列挙学校　　パート 1
列挙の基本 と 基礎的なアルゴリズム

岡本 吉央

北陸先端科学技術大学院大学

2011 年 9 月 28 日
列挙学校

列挙問題とは?

与えられた条件を満たすものを漏れなく，重複なく出力する問題
(出力順に条件のある場合もある)

列挙問題とは?

与えられた条件を満たすものを漏れなく，重複なく出力する問題
(出力順に条件のある場合もある)

列挙問題 (のインスタンス) の例

$\{1, 2, 3, 4, 5\}$ の部分集合でその和が 6 となるものを列挙せよ
答：$\{1, 2, 3\}, \{1, 5\}, \{2, 4\}$

より現実的な列挙問題 (のインスタンス) の例

「カルキ」という文字列を含む「Web ページ」を
Page ランクの大きな順に列挙せよ

列挙問題の難しさ

列挙問題 (のインスタンス) の例

$\{1,2,3,4,5\}$ の部分集合でその和が 6 となるものを列挙せよ
答：$\{1,2,3\}, \{1,5\}, \{2,4\}$

- 出力の個数 $= 3$
- $\{1,2,3,4,5\}$ の部分集合の数 $= 2^5 = 32$

次のアルゴリズムの効率は非常に悪い

- $\{1,2,3,4,5\}$ の部分集合を 1 つずつ見ていき，
  その和が 6 なら出力する

列挙問題の難しさ

列挙問題 (のインスタンス) の例

$\{1,2,3,4,5\}$ の部分集合でその和が 6 となるものを列挙せよ
答：$\{1,2,3\}, \{1,5\}, \{2,4\}$

- 出力の個数 $= 3$
- $\{1,2,3,4,5\}$ の部分集合の数 $= 2^5 = 32$

次のアルゴリズムの効率は**非常に悪い**

- $\{1,2,3,4,5\}$ の部分集合を 1 つずつ見ていき，
  その和が 6 なら出力する

効率よく 正しい 列挙をするにはどうすればよいか??

アルゴリズムの評価

アルゴリズムに対して証明すべきことは何か

- 正当性 (アルゴリズムが正しく問題を解くこと)
- 効率性 (アルゴリズムの効率がよいこと)

列挙アルゴリズムの正当性

出力すべき対象をすべて漏れなく，重複なく出力すること
(出力順指定のある問題では，その順に出力がされることも)

アルゴリズムの評価

アルゴリズムに対して証明すべきことは何か

- 正当性 (アルゴリズムが正しく問題を解くこと)
- 効率性 (アルゴリズムの効率がよいこと)

列挙アルゴリズムの効率性

理論的には「多項式時間で出力する」こと

アルゴリズムの評価

アルゴリズムに対して証明すべきことは何か

- 正当性 (アルゴリズムが正しく問題を解くこと)
- 効率性 (アルゴリズムの効率がよいこと)

列挙アルゴリズムの効率性

理論的には「多項式時間で出力する」こと

問題点

出力の個数が入力サイズに対して指数関数的かも

$\xrightarrow{\text{よって}}$ 「多項式時間で出力する」ことの意味を再考する必要あり

列挙アルゴリズムの効率の測り方 — 時間

n：入力サイズ
N：出力の個数

列挙アルゴリズムの効率の測り方 — 時間

n：入力サイズ
N：出力の個数

出力多項式時間 (output polynomial-time, polynomial total time)

n と N の多項式時間で全要素を列挙

列挙アルゴリズムの効率の測り方 — 時間

n：入力サイズ
N：出力の個数

出力多項式時間 (output polynomial-time, polynomial total time)

n と N の多項式時間で全要素を列挙

ならし多項式時間遅延 (amortized polynomial-time delay)

n に関して多項式，N に関して線形時間で全要素を列挙
(1 つの出力から次の出力までがならしで n の多項式)

列挙アルゴリズムの効率の測り方 — 時間

n：入力サイズ
N：出力の個数

出力多項式時間 (output polynomial-time, polynomial total time)

n と N の多項式時間で全要素を列挙

ならし多項式時間遅延 (amortized polynomial-time delay)

n に関して多項式，N に関して線形時間で全要素を列挙
(1 つの出力から次の出力までがならしで n の多項式)

最悪時多項式時間遅延 (worst-case polynomial-time delay)

1 つの出力から次の出力までが n の多項式

遅延を考える場合，次の 2 つも n の多項式でないといけない
前処理時間：アルゴリズム開始から最初の出力までの時間
後処理時間：最後の出力からアルゴリズム停止までの時間

効率の間の関係

観察

アルゴリズムの計算量が出力多項式時間
$\Leftarrow$ ならし多項式時間遅延
$\Leftarrow$ 最悪時多項式時間遅延

例

総時間	出力多項式	ならし多項式遅延	最悪時多項式遅延
$O(n^3N^2)$	○	×	×
$O(n^4N)$	○	○	×

注：総時間を見ただけでは最悪時多項式時間遅延かどうか不明

列挙アルゴリズムの効率の測り方 — 領域

n：入力サイズ
N：出力の個数

多項式領域 (polynomial space)

n の多項式領域で全要素を列挙

「領域計算量」を考えるときは
作業テープの領域を測り，出力テープの領域は測らない
(出力テープの領域使用は $\Omega(N)$)

効率的な列挙アルゴリズムを設計することの難しさ

重複をなくそうとすると...

- すべての出力を作業テープ上にとっておけばよい
  $\xrightarrow{\text{それだと}}$ 多項式領域アルゴリズムにならない (ことが多い)
- すべての出力を作業テープ上にとっておけない

効率的な列挙アルゴリズムを設計することの難しさ

重複をなくそうとすると...

- すべての出力を作業テープ上にとっておけばよい
  $\xrightarrow{\text{それだと}}$ 多項式領域アルゴリズムにならない (ことが多い)
- すべての出力を作業テープ上にとっておけない

加えて，漏れをなくそうとすると...

- 出力するべき対象の数が最初から分かっていればよい
  $\xrightarrow{\text{しかし}}$ その数を前もって計算することが難しい (ことが多い)

  (cf. #P **困難性**)
- 数さえ分からないので，いつ停止すればよいか分からない

(マラソン計時の例)

効率的な列挙アルゴリズムを設計することの難しさ

重複をなくそうとすると...

- すべての出力を作業テープ上にとっておけばよい
  $\xrightarrow{\text{それだと}}$ 多項式領域アルゴリズムにならない (ことが多い)
- すべての出力を作業テープ上にとっておけない

加えて，漏れをなくそうとすると...

- 出力するべき対象の数が最初から分かっていればよい
  $\xrightarrow{\text{しかし}}$ その数を前もって計算することが難しい (ことが多い)

  (cf. #P **困難性**)
- 数さえ分からないので，いつ停止すればよいか分からない

(マラソン計時の例)

複雑な対象を効率よく列挙することは夢のことのように思える

効率的な列挙アルゴリズムを設計することの難しさ

重複をなくそうとすると...

- すべての出力を作業テープ上にとっておけばよい
  $\xrightarrow{\text{それだと}}$ 多項式領域アルゴリズムにならない (ことが多い)
- すべての出力を作業テープ上にとっておけない

加えて，漏れをなくそうとすると...

- 出力するべき対象の数が最初から分かっていればよい
  $\xrightarrow{\text{しかし}}$ その数を前もって計算することが難しい (ことが多い)

  (cf. #P **困難性**)
- 数さえ分からないので，いつ停止すればよいか分からない

(マラソン計時の例)

複雑な対象を効率よく列挙することは夢のことのように思える
けど，できる場合もある!!

パート 1 の内容

- 列挙問題，列挙アルゴリズムとは?
- 列挙問題，列挙アルゴリズム設計の難しさ
- 列挙アルゴリズム設計技法
  - 分割探索 (binary partition)
  - 組合せ Gray 符号 (combinatorial Gray code)
  - 逆探索 (reverse search)
- 難しい列挙問題

以降は

- 中野眞一先生
  グラフ列挙 (パート 1 より複雑な対象の列挙)
- 有村博紀先生
  パターンマイニング (もっと複雑な対象の列挙)
- 宇野毅明先生
  複雑な構造の列挙 (もっともっと複雑な対象の列挙)

部分集合列挙問題

入力：自然数 n
出力：集合 $\{1, 2, \ldots, n\}$ の部分集合すべて

例：入力 n が 4 の場合の出力

$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$
$\{4\}$	$\{1,2\}$	$\{1,3\}$	$\{1,4\}$
$\{2,3\}$	$\{2,4\}$	$\{3,4\}$	$\{1,2,3\}$
$\{1,2,4\}$	$\{1,3,4\}$	$\{2,3,4\}$	$\{1,2,3,4\}$

ややこしい注意 (アルゴリズムに十分慣れてる方だけのために)

計算モデルは word RAM を仮定．入力自然数は (16 ビットや 32 ビットとは限らない) 1 語に収まり，語に対する操作は定数時間でできるものとする．領域計算量も語数を数える．また，入力 n は 1 進符号化 (unary encoded) されていると仮定する．

分割探索で部分集合列挙問題を解く

入力：自然数 n

アルゴリズム設計方針

場合分けする

1. **問題を**
   - **「1」を含む部分集合すべてを出力する問題**
   - **「1」を含まない部分集合すべてを出力する問題**

   に (仮想的に) 分割
2. **そのそれぞれを**
   - **「2」を含む部分集合すべてを出力する問題**
   - **「2」を含まない部分集合すべてを出力する問題**

   に (仮想的に) 分割
3. **そのそれぞれを....**

アルゴリズムの動作を見てみる

$\emptyset$	$\{4\}$	$\{1\}$	$\{1,4\}$
$\{3\}$	$\{3,4\}$	$\{1,3\}$	$\{1,3,4\}$
$\{2\}$	$\{2,4\}$	$\{1,2\}$	$\{1,2,4\}$
$\{2,3\}$	$\{2,3,4\}$	$\{1,2,3\}$	$\{1,2,3,4\}$

アルゴリズムの動作を見てみる

$\emptyset$	$\{4\}$	$\{1\}$	$\{1,4\}$
$\{3\}$	$\{3,4\}$	$\{1,3\}$	$\{1,3,4\}$
$\{2\}$	$\{2,4\}$	$\{1,2\}$	$\{1,2,4\}$
$\{2,3\}$	$\{2,3,4\}$	$\{1,2,3\}$	$\{1,2,3,4\}$

アルゴリズムの動作を見てみる

$\emptyset$	$\{4\}$	$\{1\}$	$\{1,4\}$
$\{3\}$	$\{3,4\}$	$\{1,3\}$	$\{1,3,4\}$
$\{2\}$	$\{2,4\}$	$\{1,2\}$	$\{1,2,4\}$
$\{2,3\}$	$\{2,3,4\}$	$\{1,2,3\}$	$\{1,2,3,4\}$

アルゴリズムの動作を見てみる

$\emptyset$	$\{4\}$	$\{1\}$	$\{1,4\}$
$\{3\}$	$\{3,4\}$	$\{1,3\}$	$\{1,3,4\}$
$\{2\}$	$\{2,4\}$	$\{1,2\}$	$\{1,2,4\}$
$\{2,3\}$	$\{2,3,4\}$	$\{1,2,3\}$	$\{1,2,3,4\}$

アルゴリズムの動作を見てみる

$\emptyset$	$\{4\}$	$\{1\}$	$\{1,4\}$
$\{3\}$	$\{3,4\}$	$\{1,3\}$	$\{1,3,4\}$
$\{2\}$	$\{2,4\}$	$\{1,2\}$	$\{1,2,4\}$
$\{2,3\}$	$\{2,3,4\}$	$\{1,2,3\}$	$\{1,2,3,4\}$

部分集合列挙アルゴリズム (分割探索)

入力：自然数 n
出力： $\{1, \ldots, n\}$ **の部分集合すべて**

- $\mathrm{A}(\emptyset, 1)$ **を呼び出し**

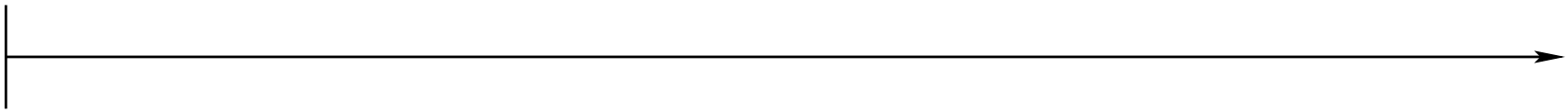
$\mathrm{A}(X, i)$

事前条件： $i \in \{1, \ldots, n, n{+}1\}$, $X \subseteq \{1, \ldots, i{-}1\}$
事後条件： $\{X \cup Y \mid Y \subseteq \{i, \ldots, n\}\}$ **の要素をすべて出力**

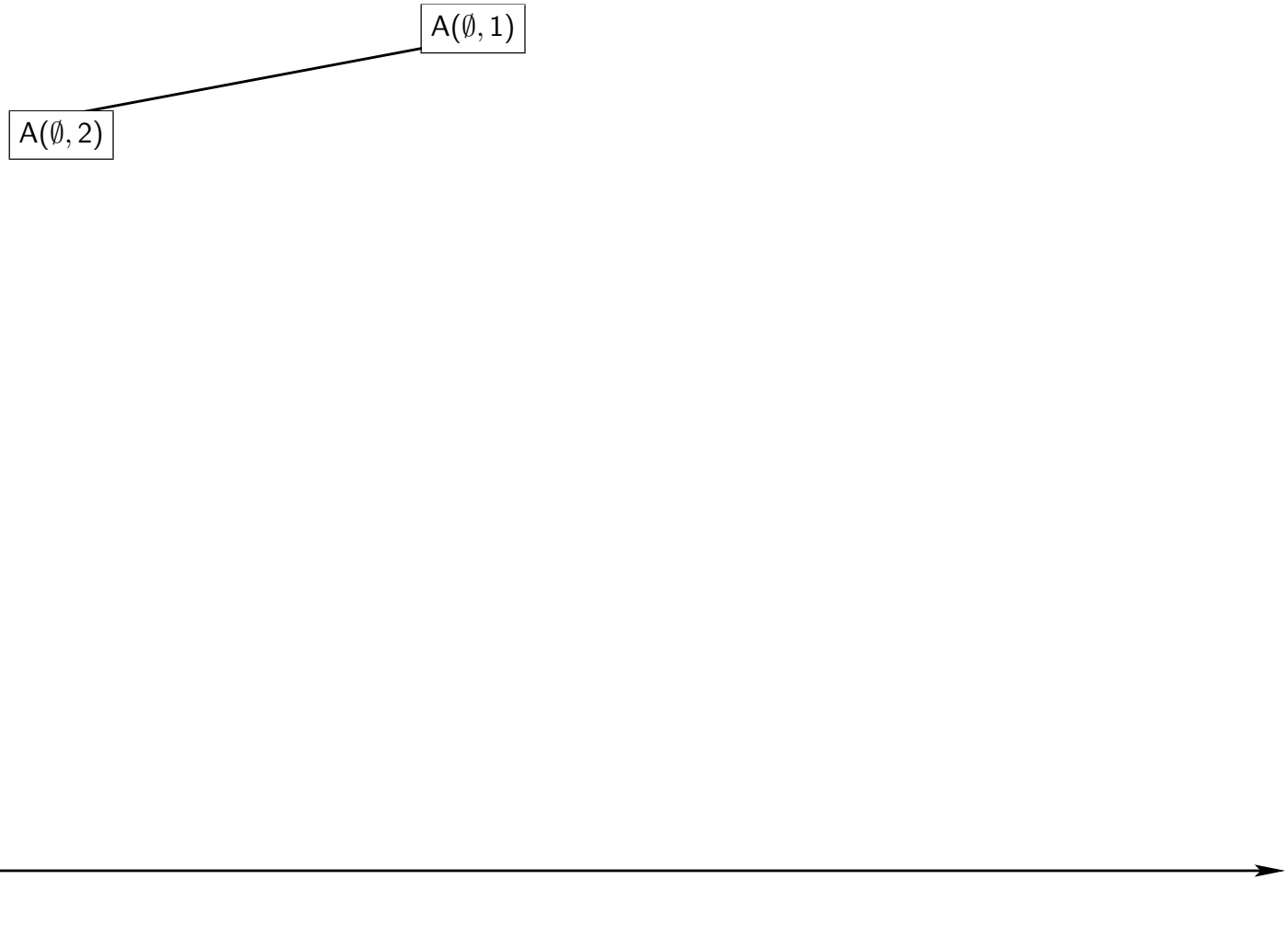
- $i = n{+}1$ **ならば，** X **を出力して終了**
- **そうでなければ，** $\mathrm{A}(X, i{+}1)$ **と** $\mathrm{A}(X \cup \{i\}, i{+}1)$ **を呼び出し**

詳細版の動きを見てみる

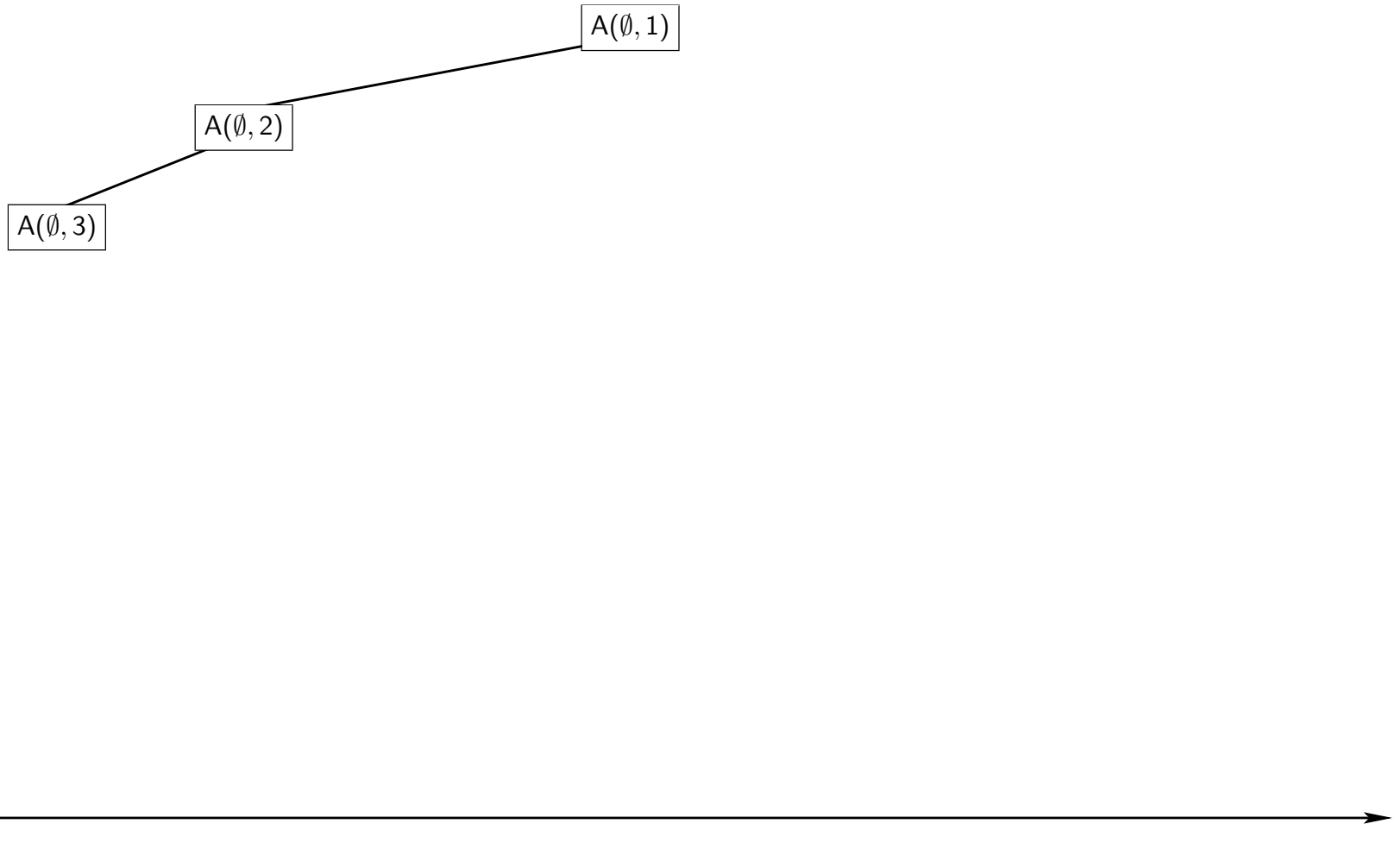
$A(\emptyset, 1)$



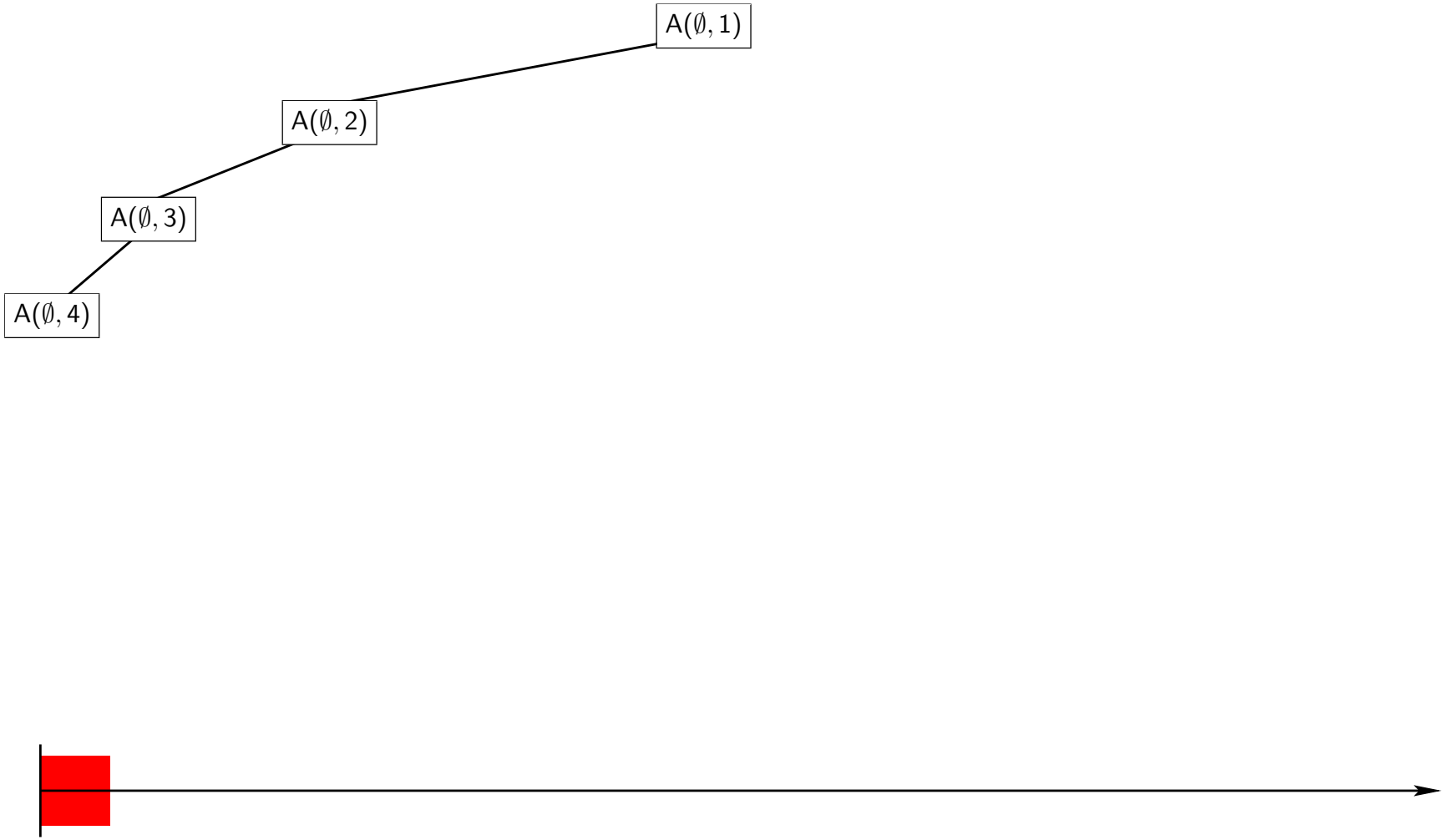
詳細版の動きを見てみる



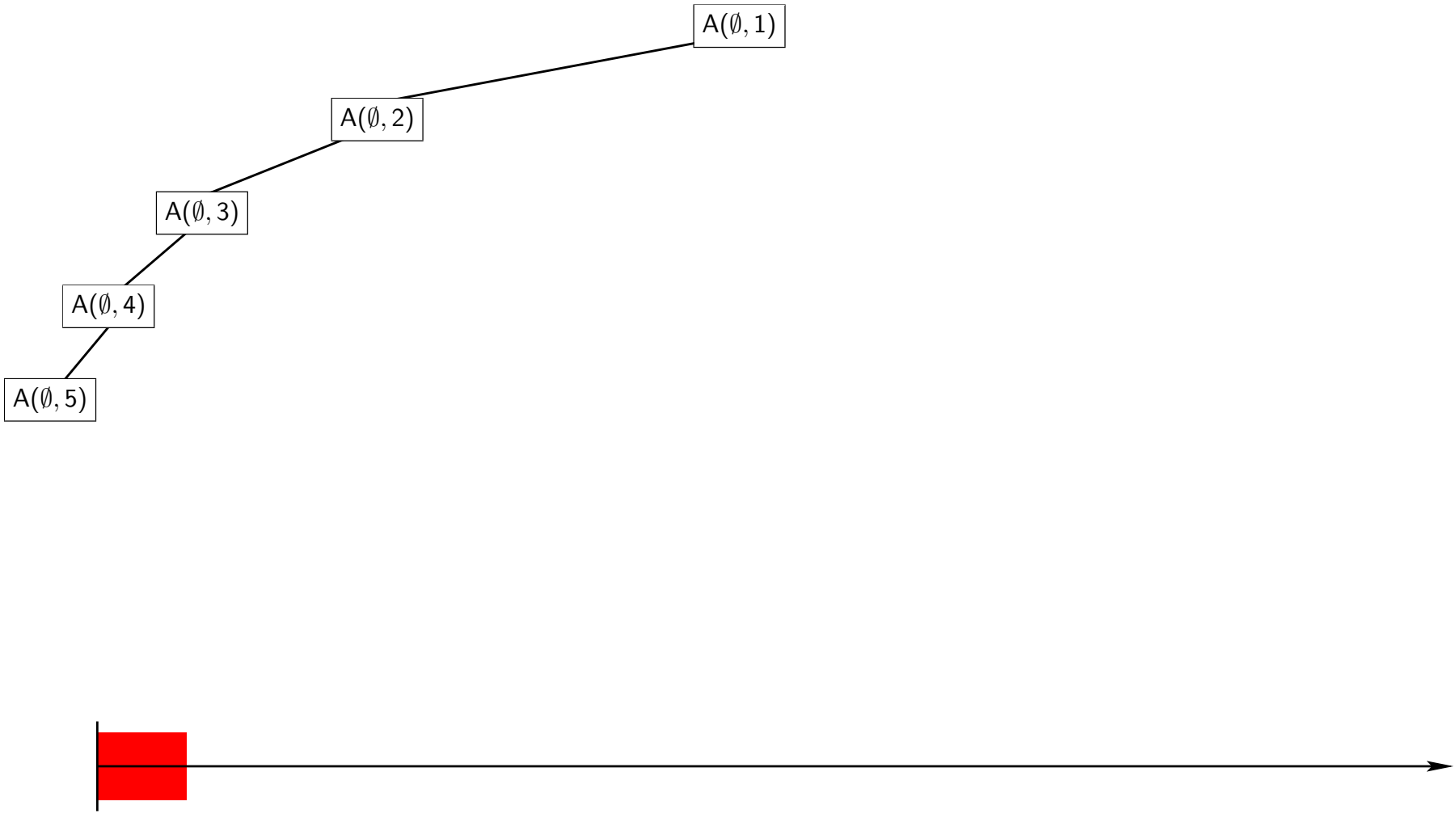
詳細版の動きを見てみる



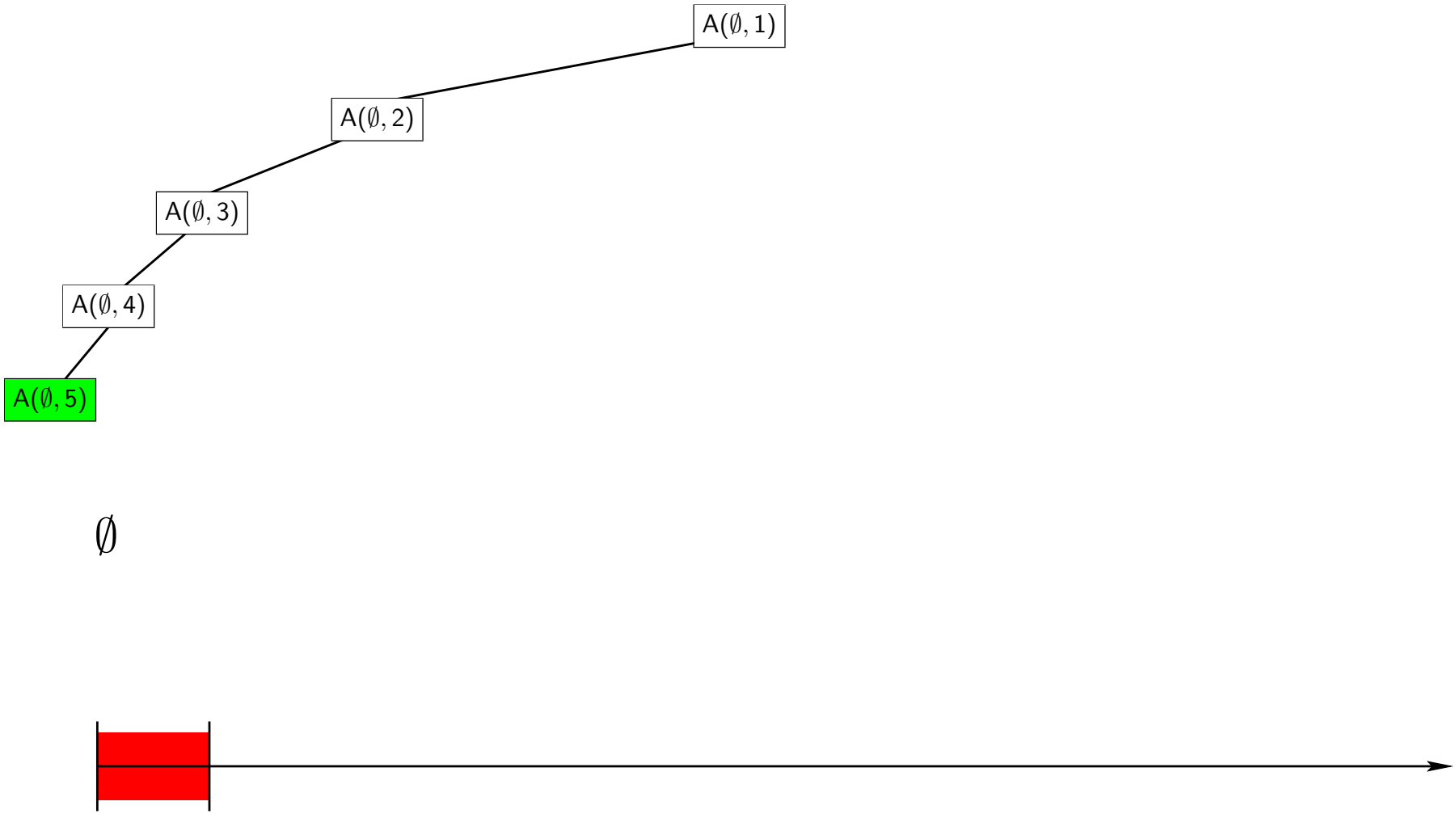
詳細版の動きを見てみる



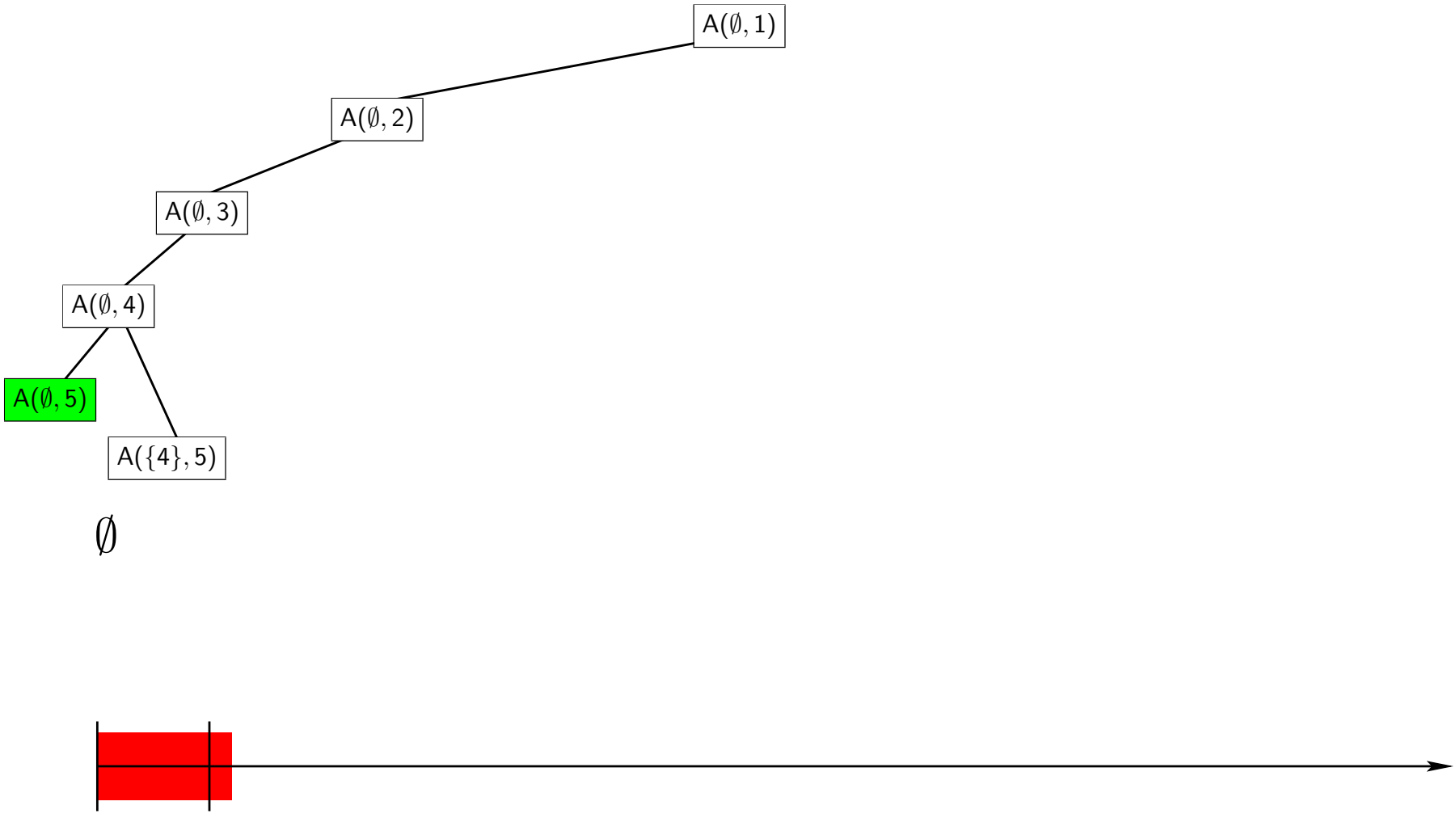
詳細版の動きを見てみる



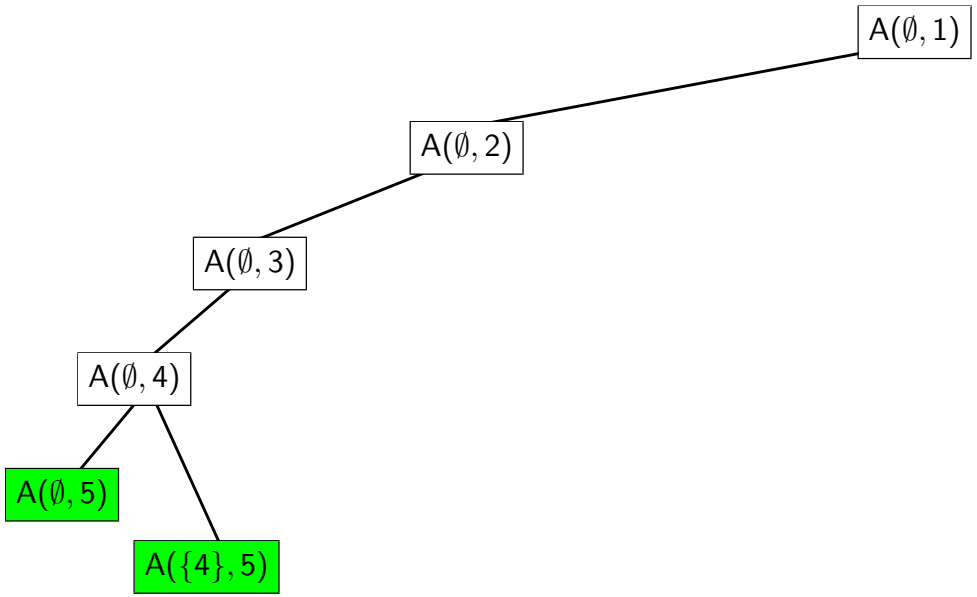
詳細版の動きを見てみる



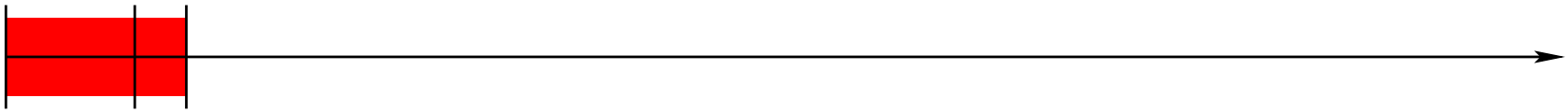
詳細版の動きを見てみる



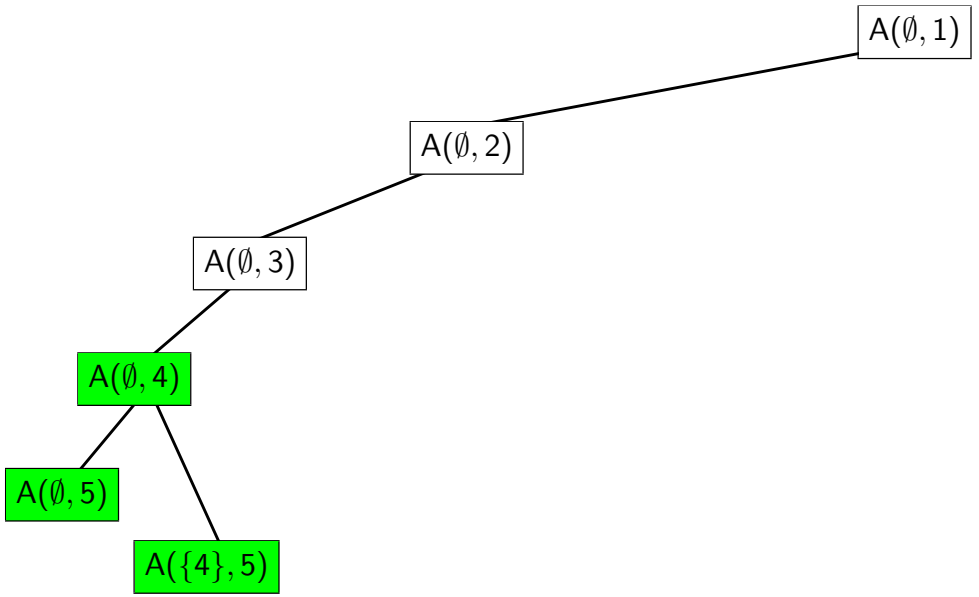
詳細版の動きを見てみる



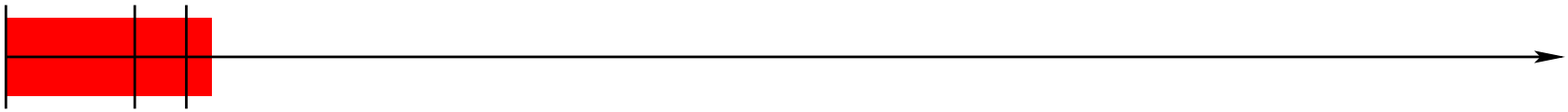
$\emptyset$, $\{4\}$



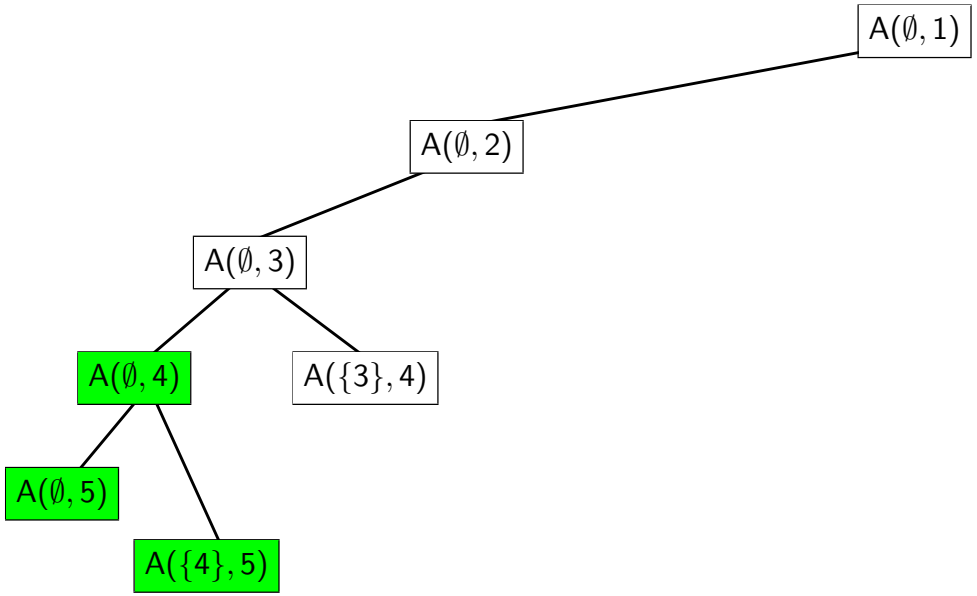
詳細版の動きを見てみる



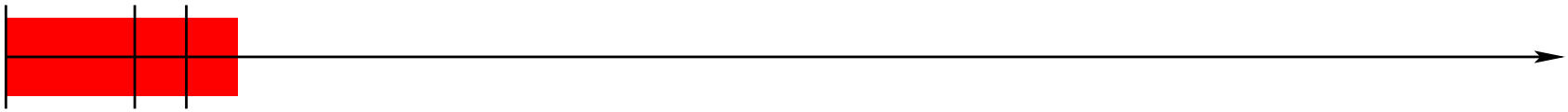
$\emptyset, \{4\}$



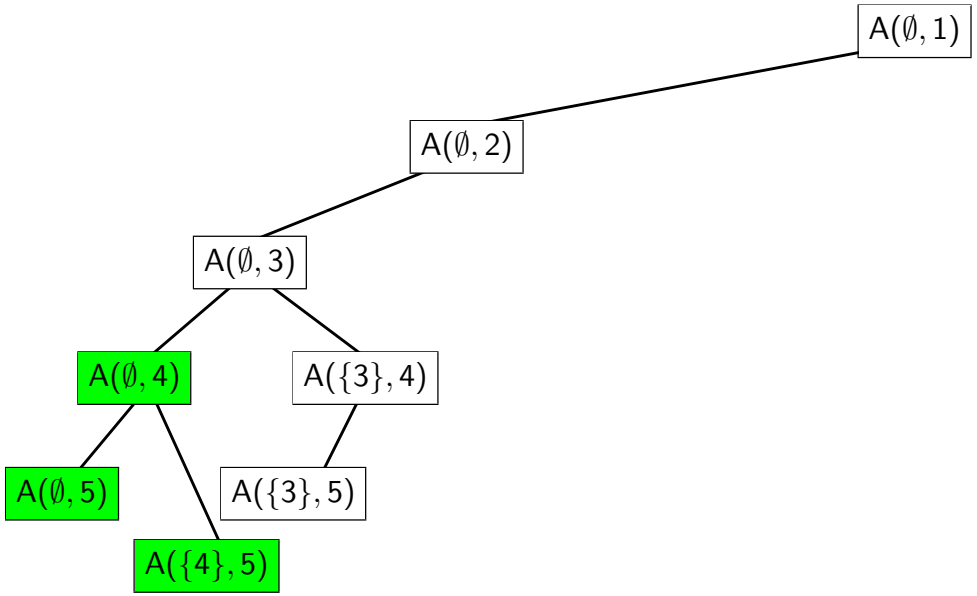
詳細版の動きを見てみる



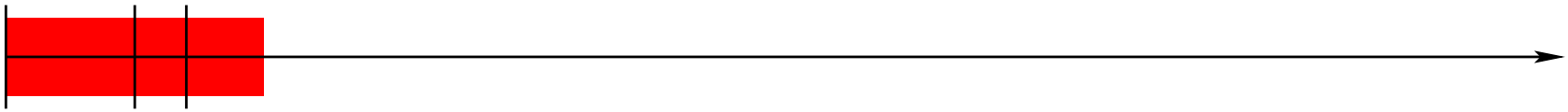
$\emptyset$, $\{4\}$



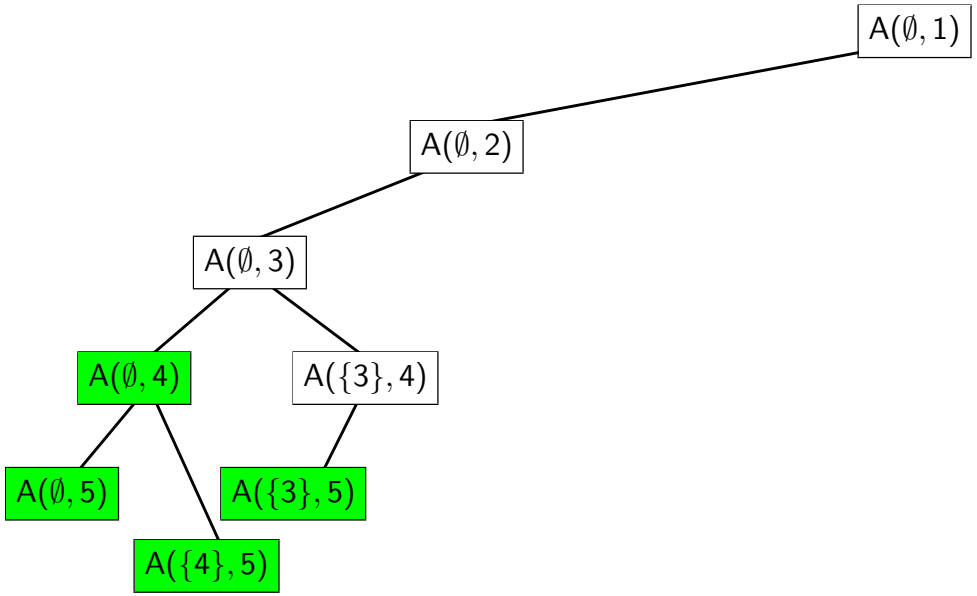
詳細版の動きを見てみる



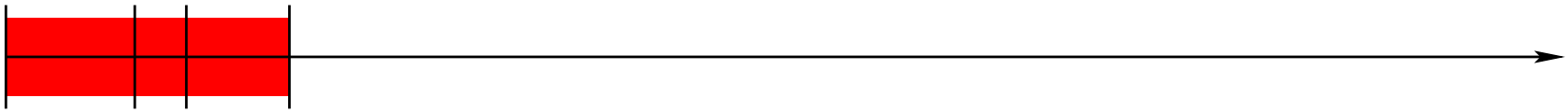
$\emptyset, \{4\}$



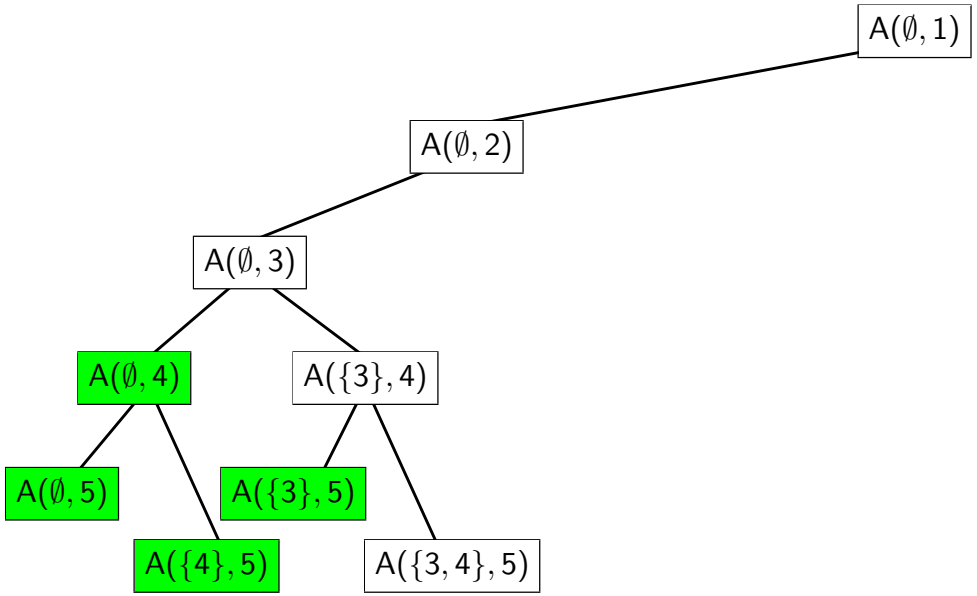
詳細版の動きを見てみる



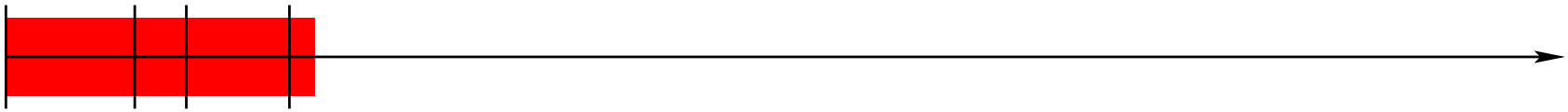
$\emptyset$, $\{4\}$, $\{3\}$



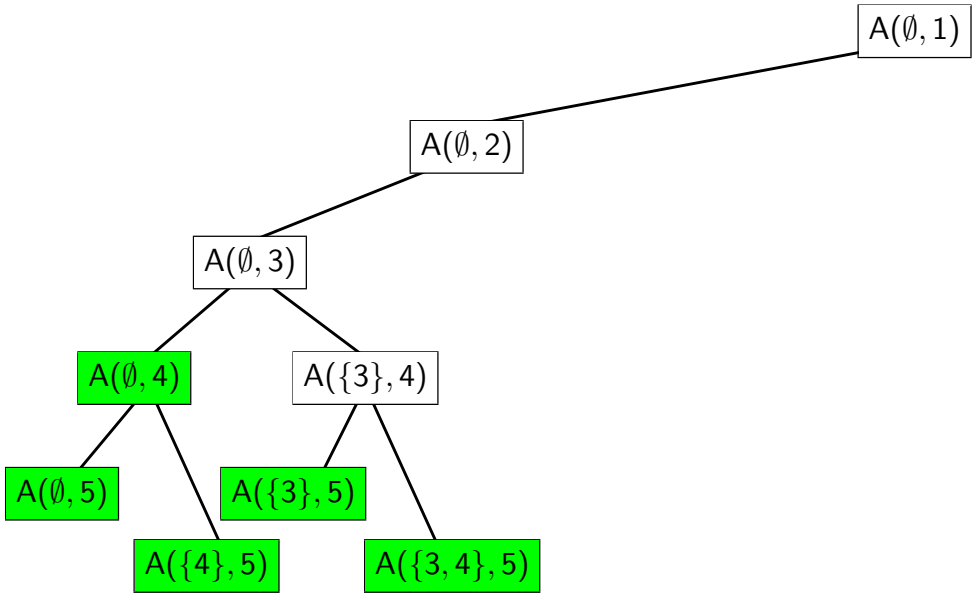
詳細版の動きを見てみる



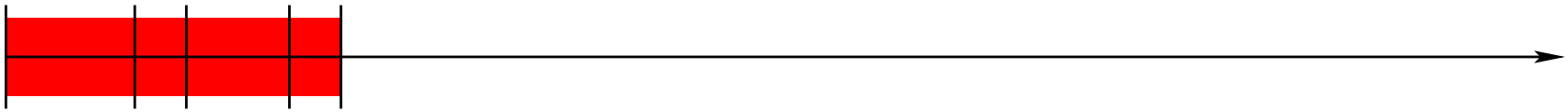
$\emptyset$, $\{4\}$, $\{3\}$



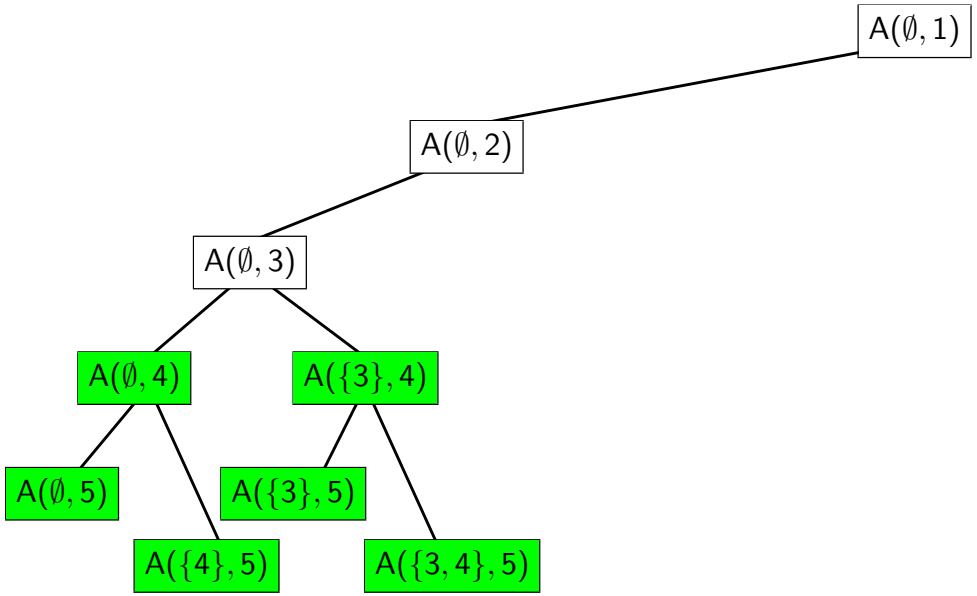
詳細版の動きを見てみる



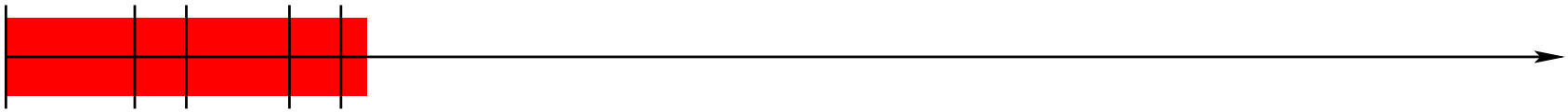
$\emptyset$, $\{4\}$, $\{3\}$, $\{3, 4\}$



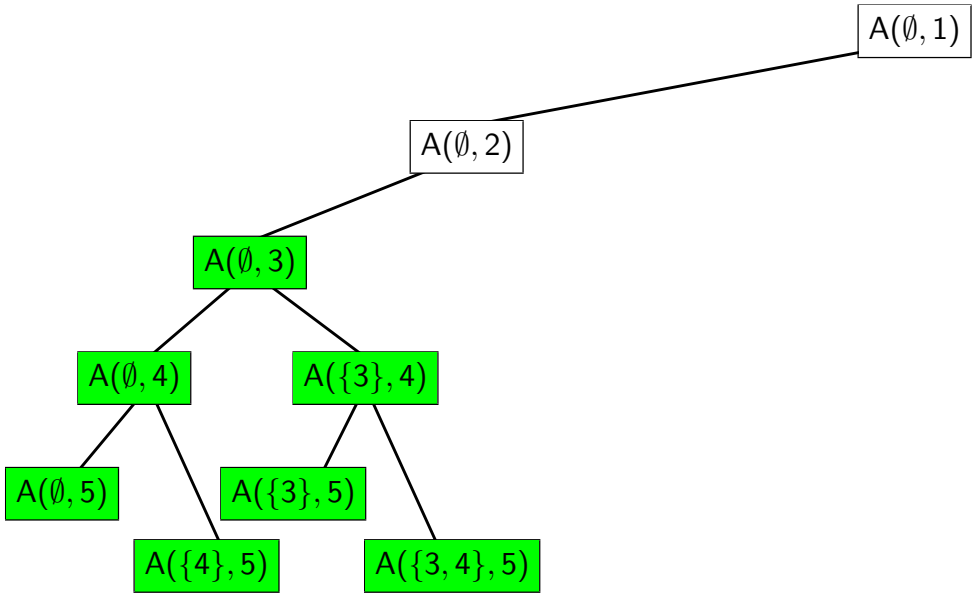
詳細版の動きを見てみる



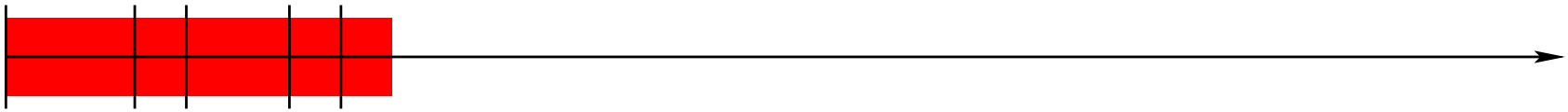
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$



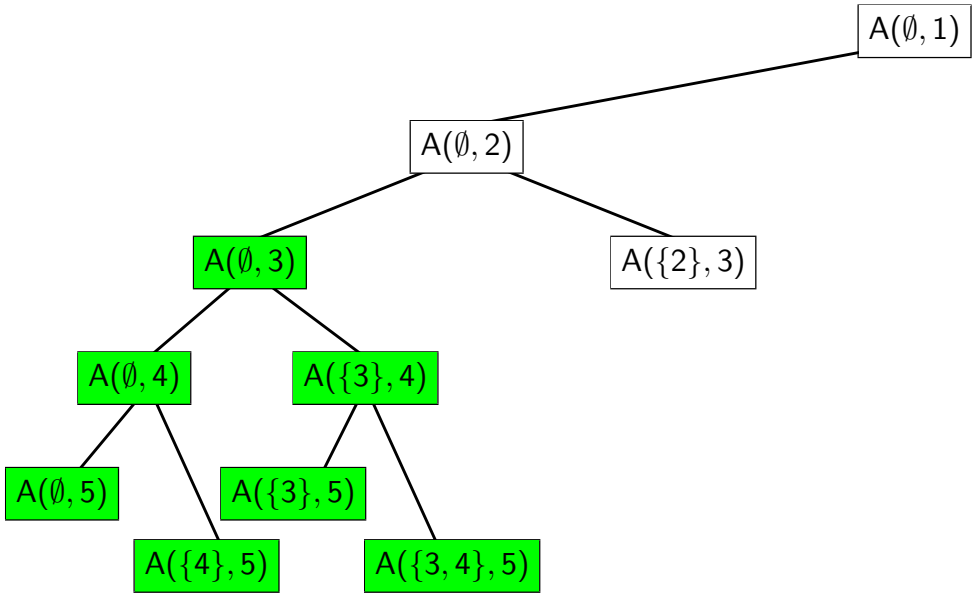
詳細版の動きを見てみる



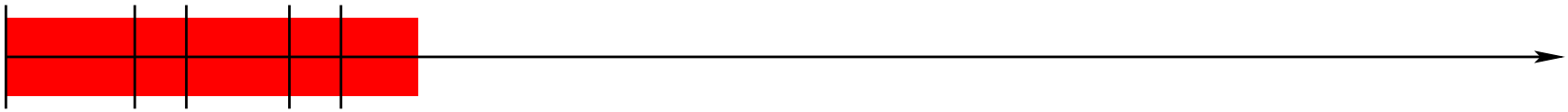
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$



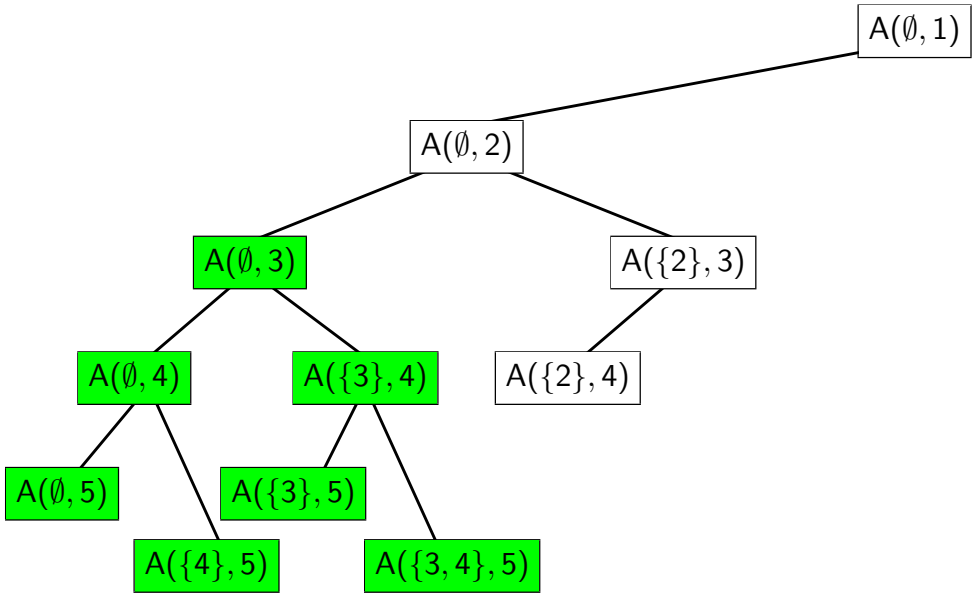
詳細版の動きを見てみる



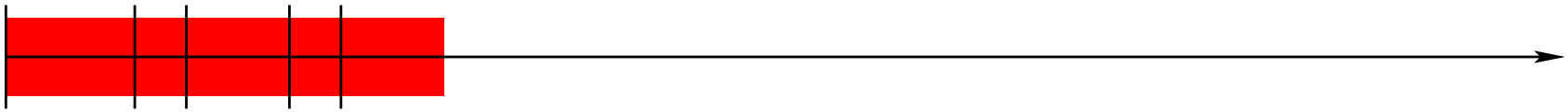
$\emptyset$, $\{4\}$, $\{3\}$, $\{3, 4\}$



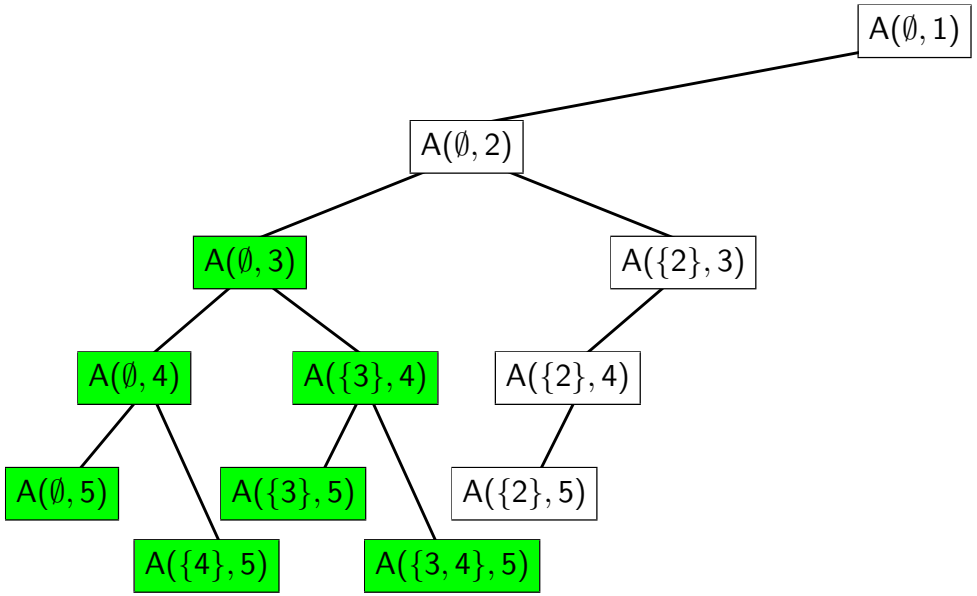
詳細版の動きを見てみる



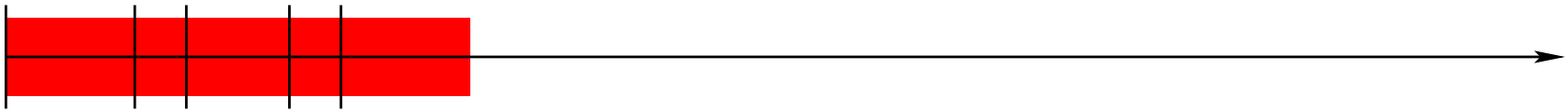
$\emptyset$, $\{4\}$, $\{3\}$, $\{3, 4\}$



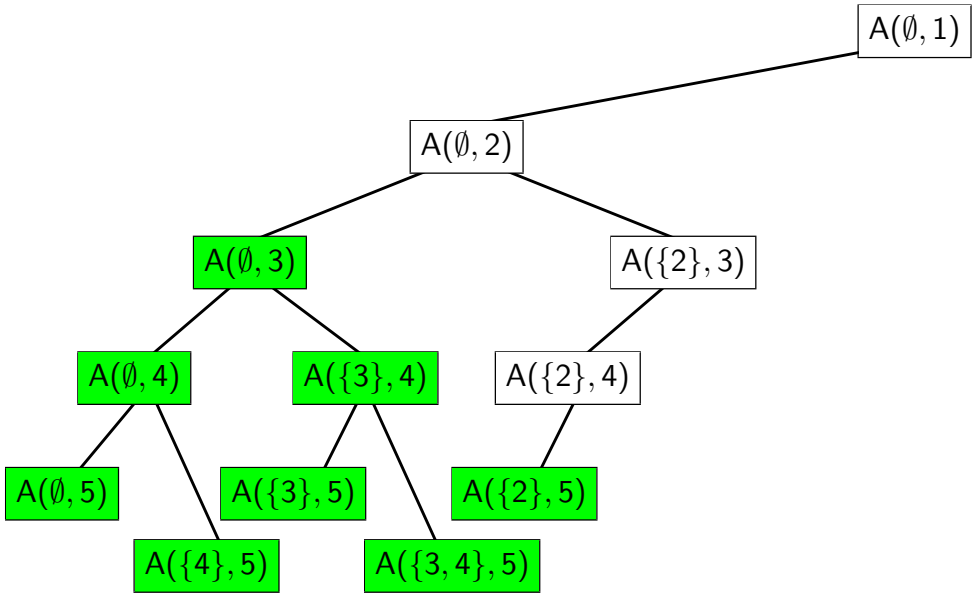
詳細版の動きを見てみる



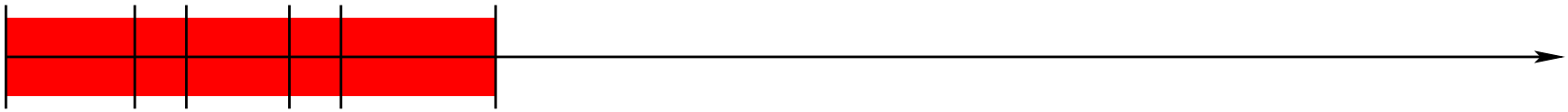
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$



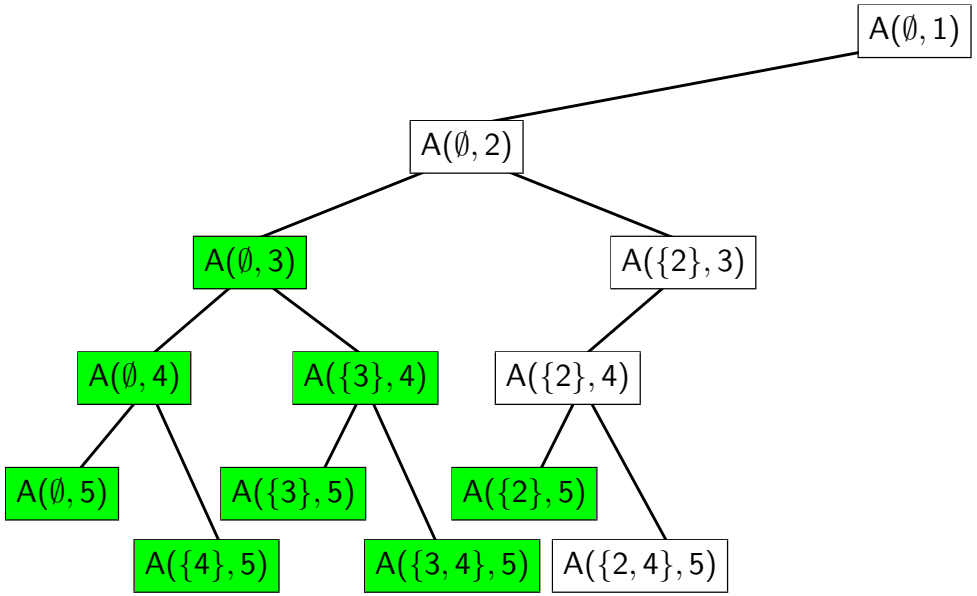
詳細版の動きを見てみる



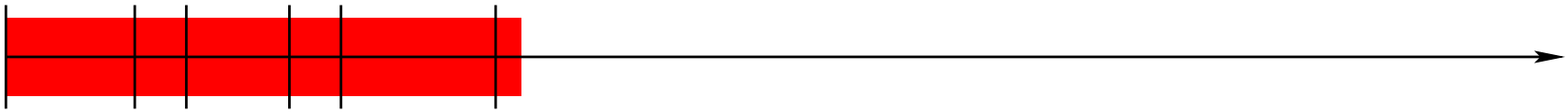
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$



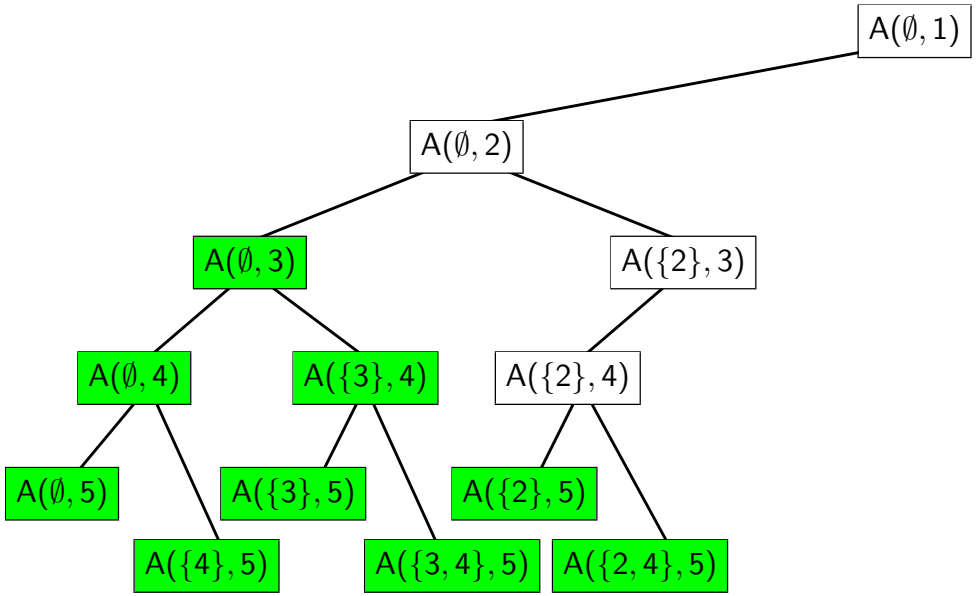
詳細版の動きを見てみる



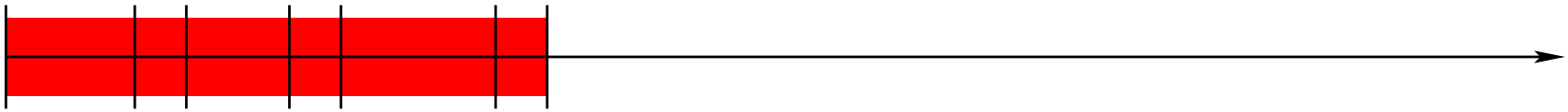
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$



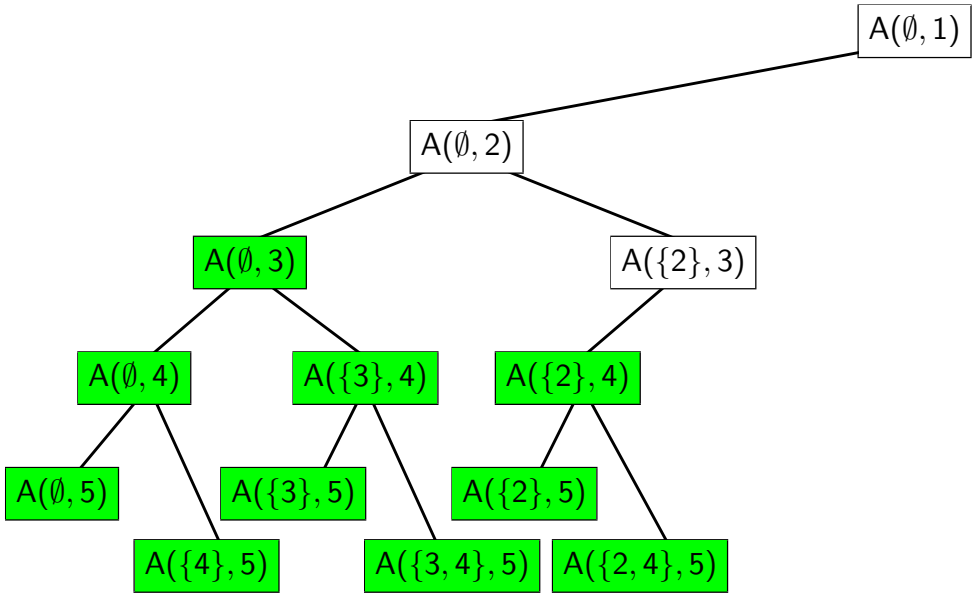
詳細版の動きを見てみる



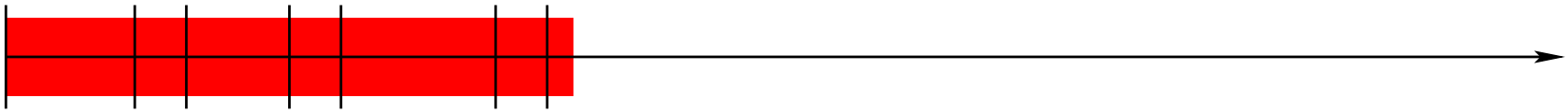
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$



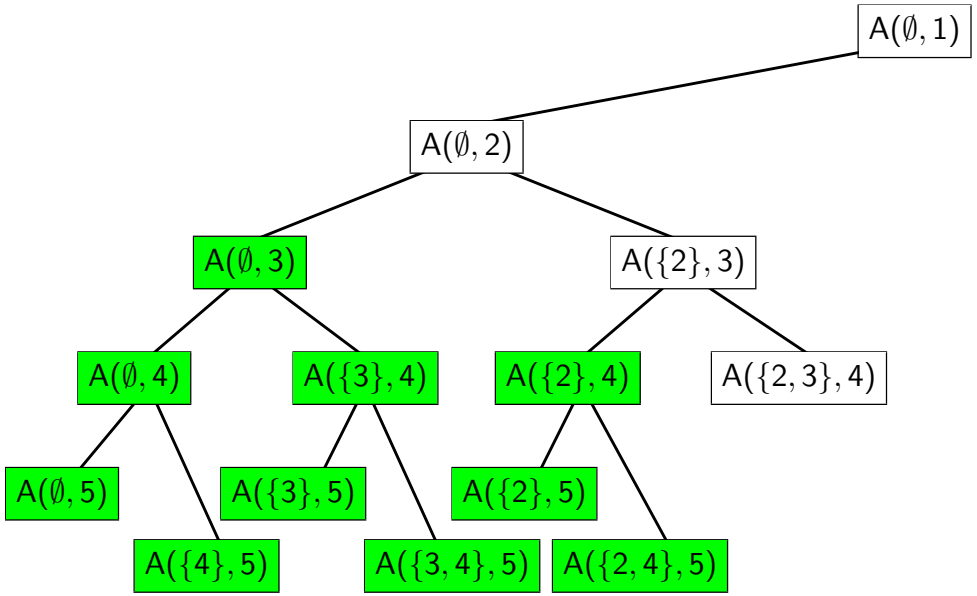
詳細版の動きを見てみる



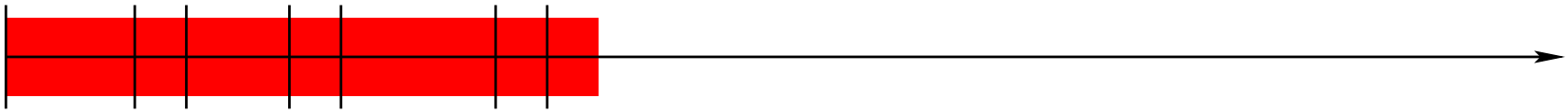
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$



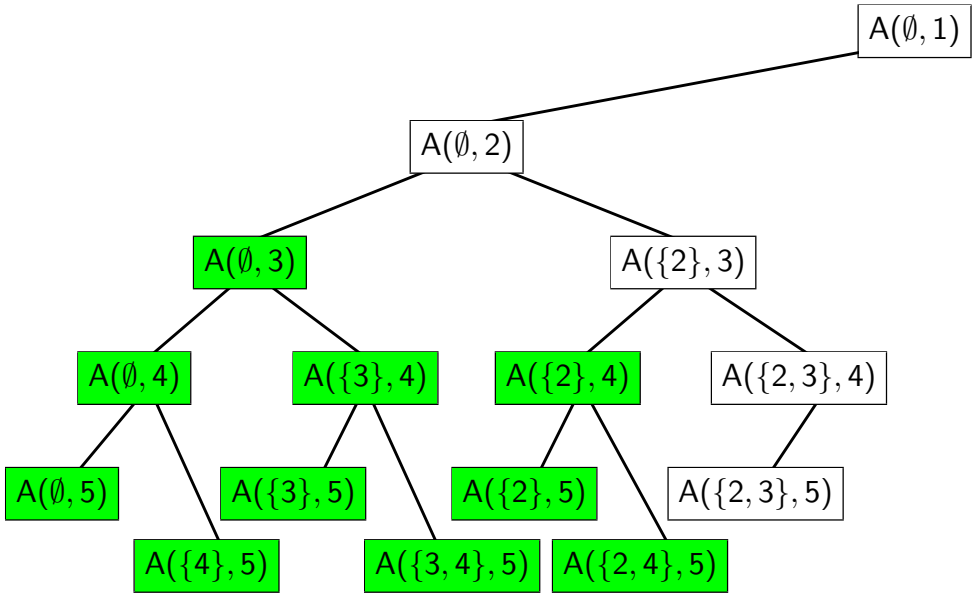
詳細版の動きを見てみる



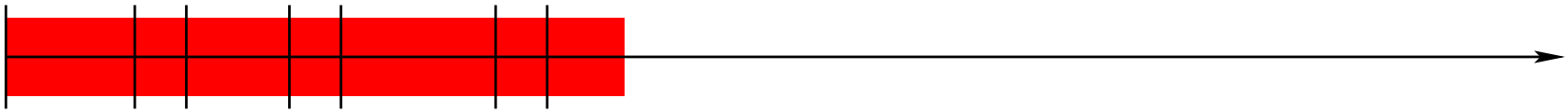
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$



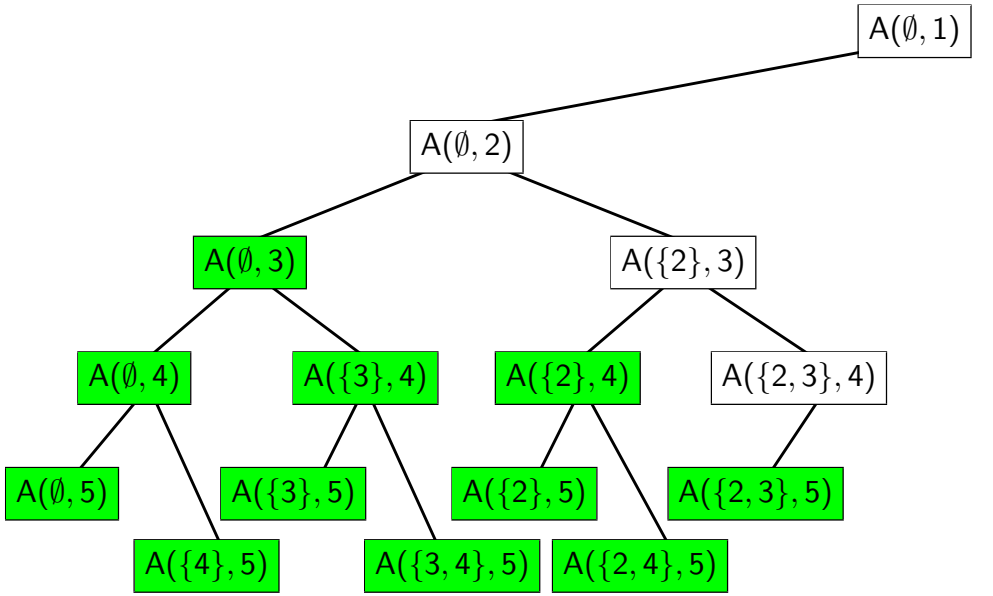
詳細版の動きを見てみる



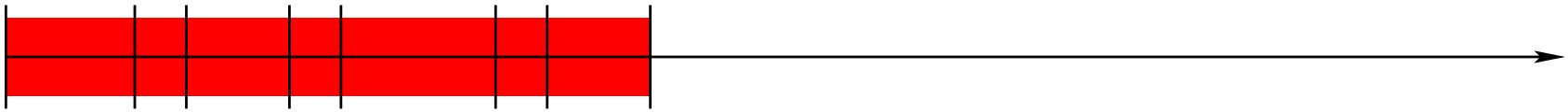
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$



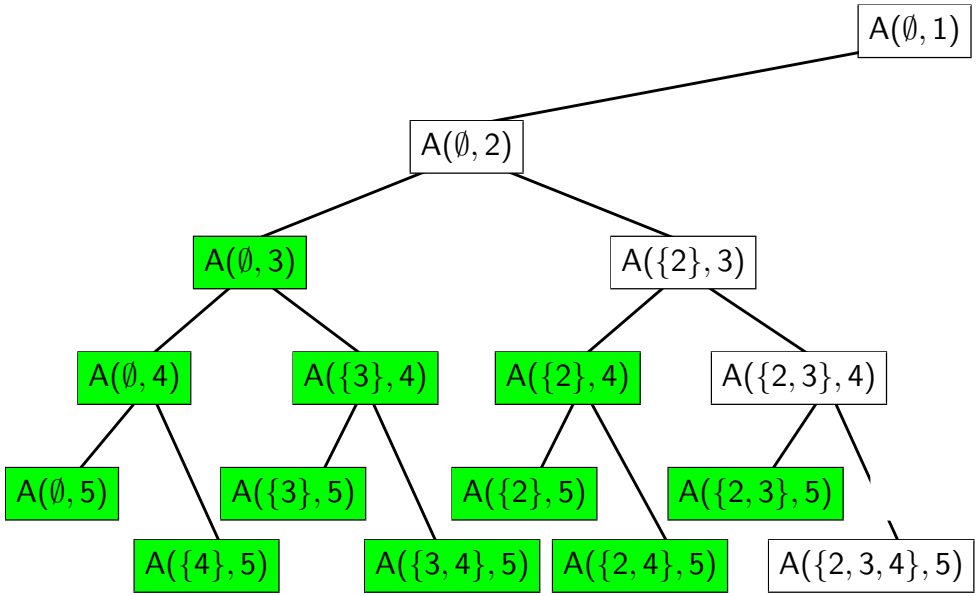
詳細版の動きを見てみる



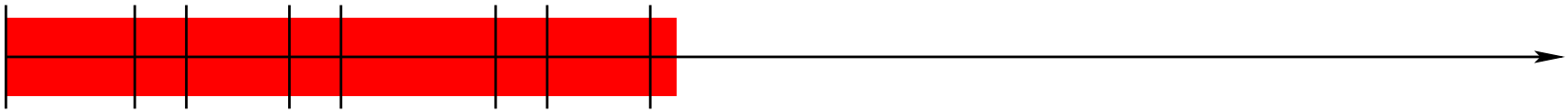
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$



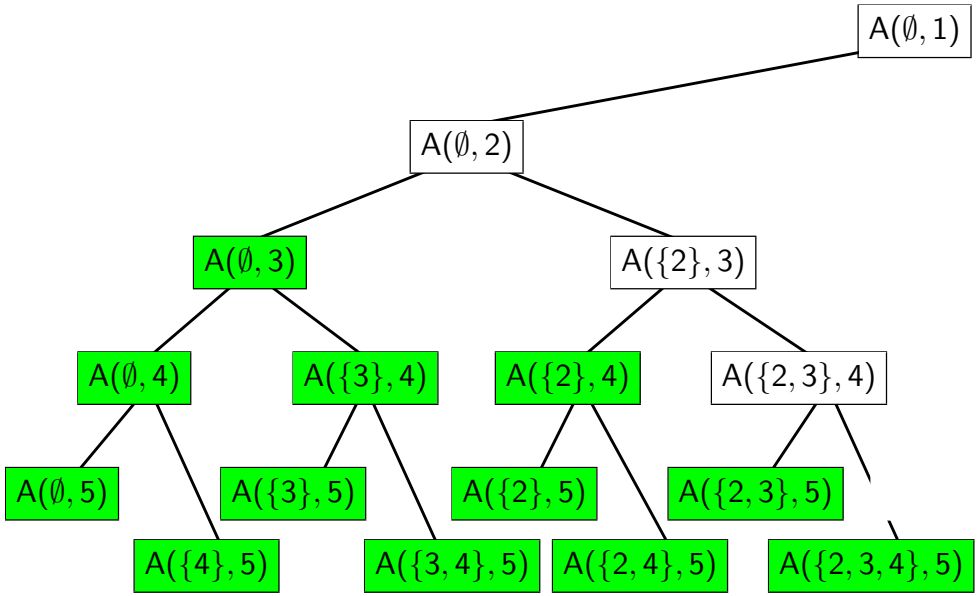
詳細版の動きを見てみる



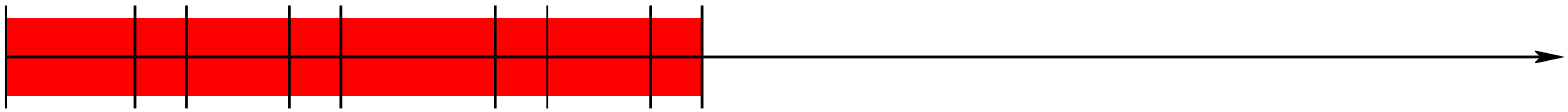
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$



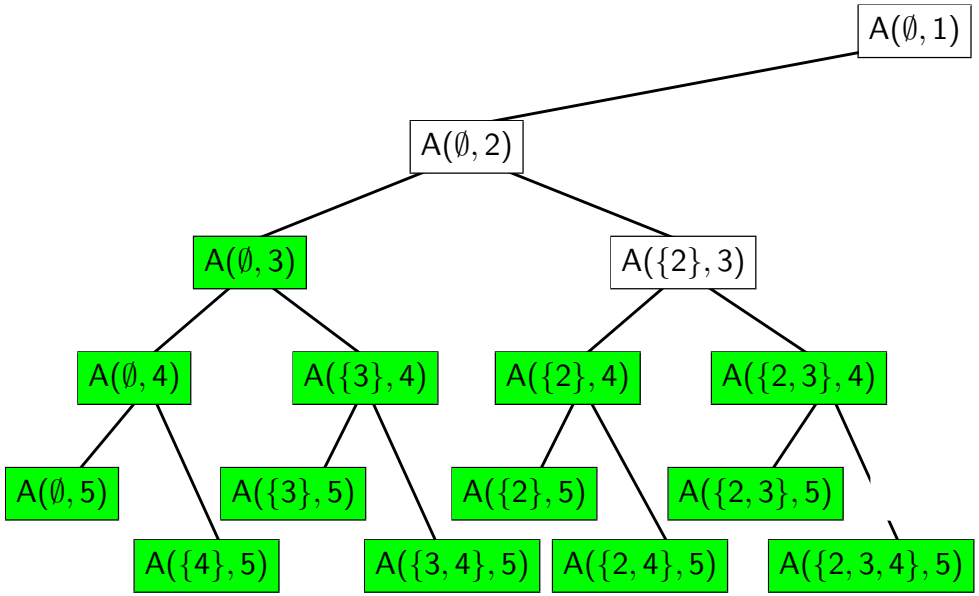
詳細版の動きを見てみる



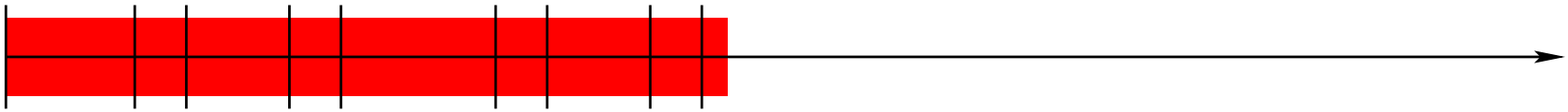
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$



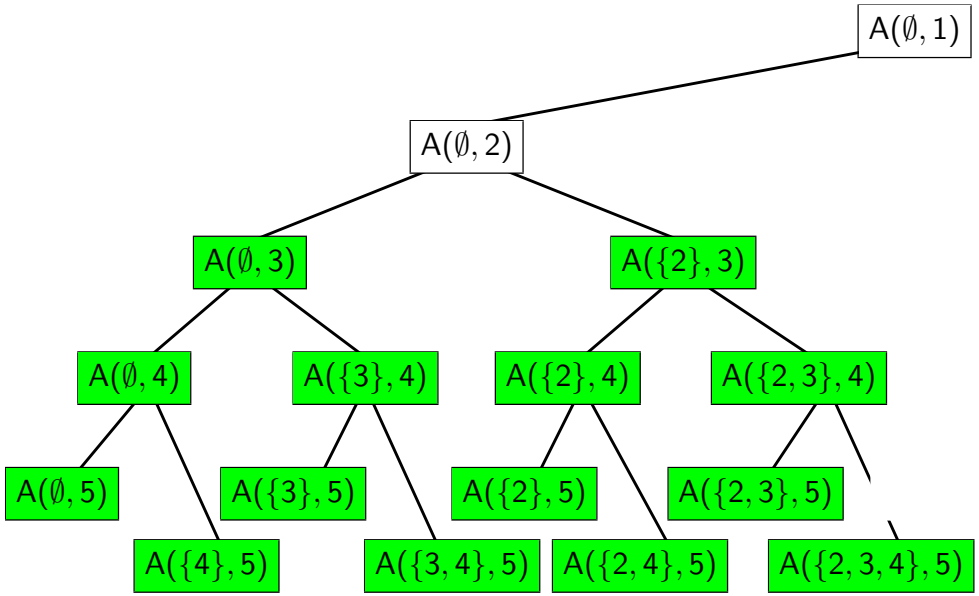
詳細版の動きを見てみる



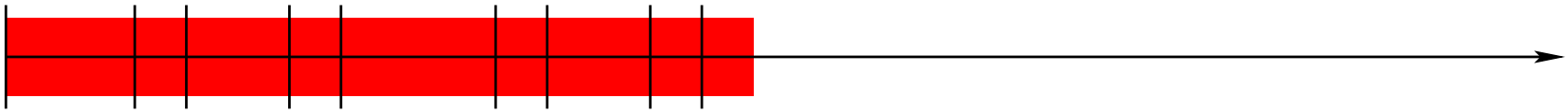
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$



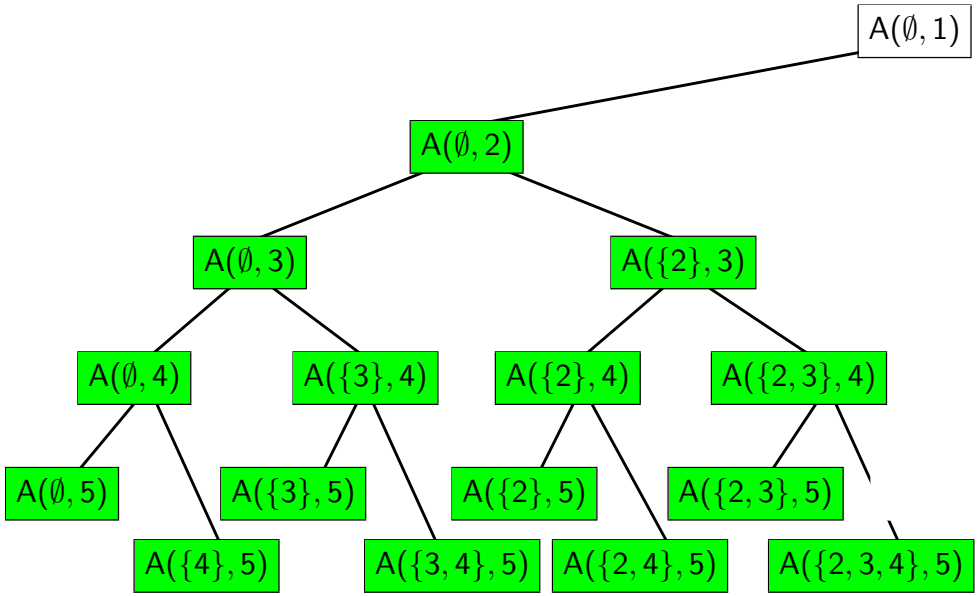
詳細版の動きを見てみる



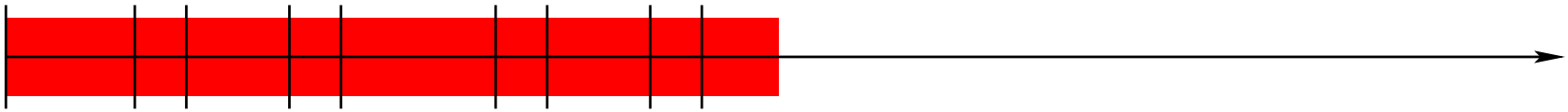
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$



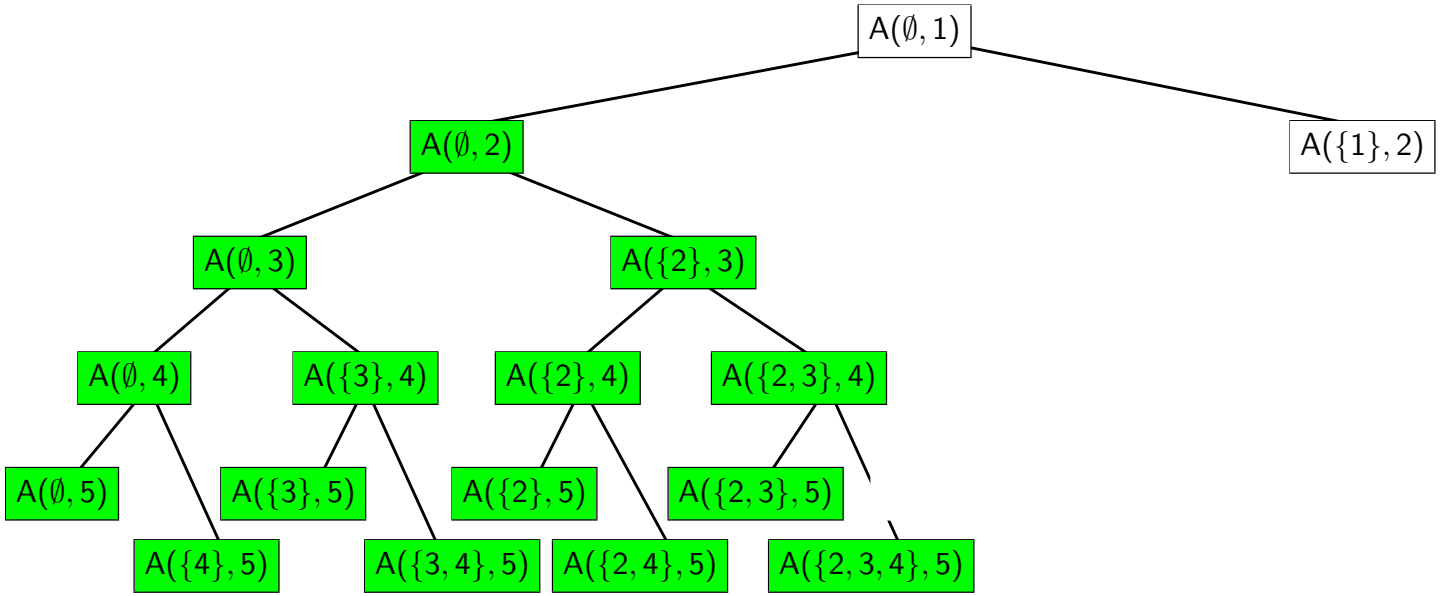
詳細版の動きを見てみる



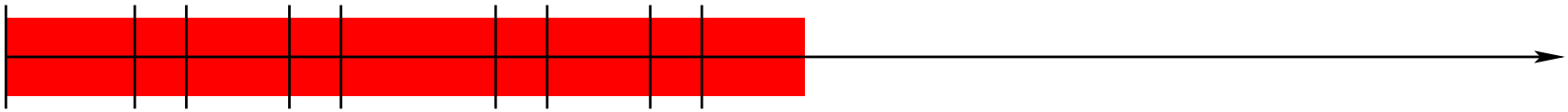
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$



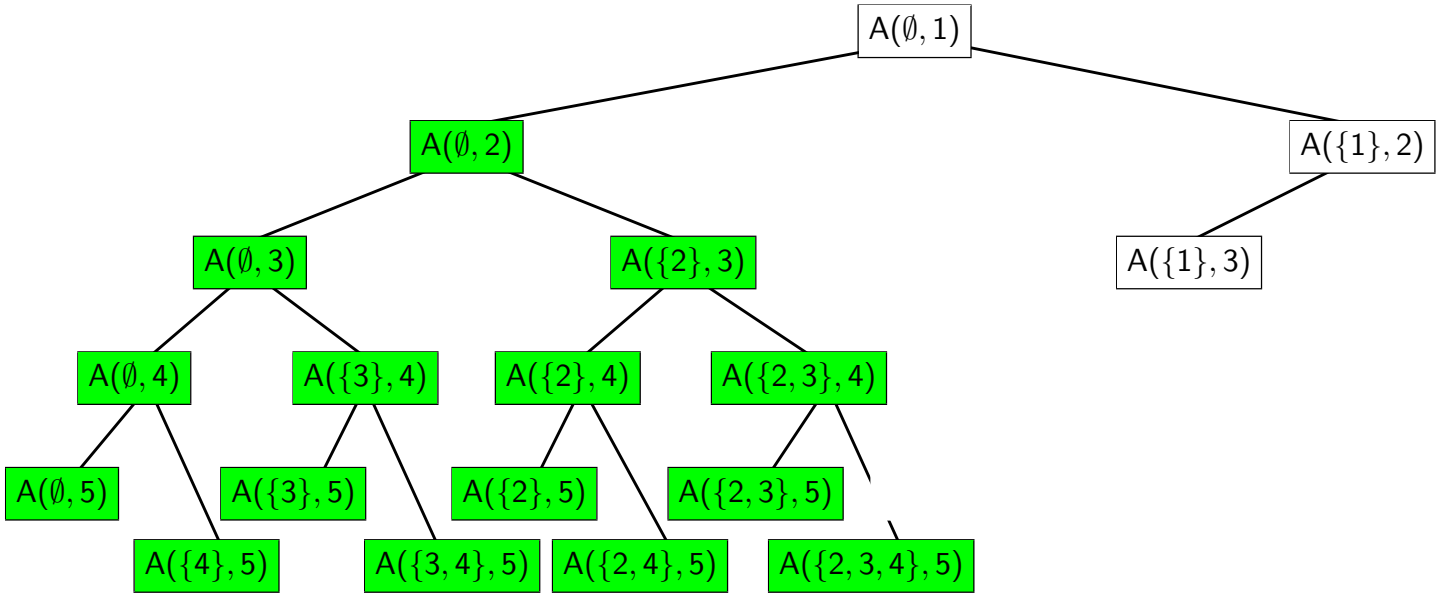
詳細版の動きを見てみる



$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$

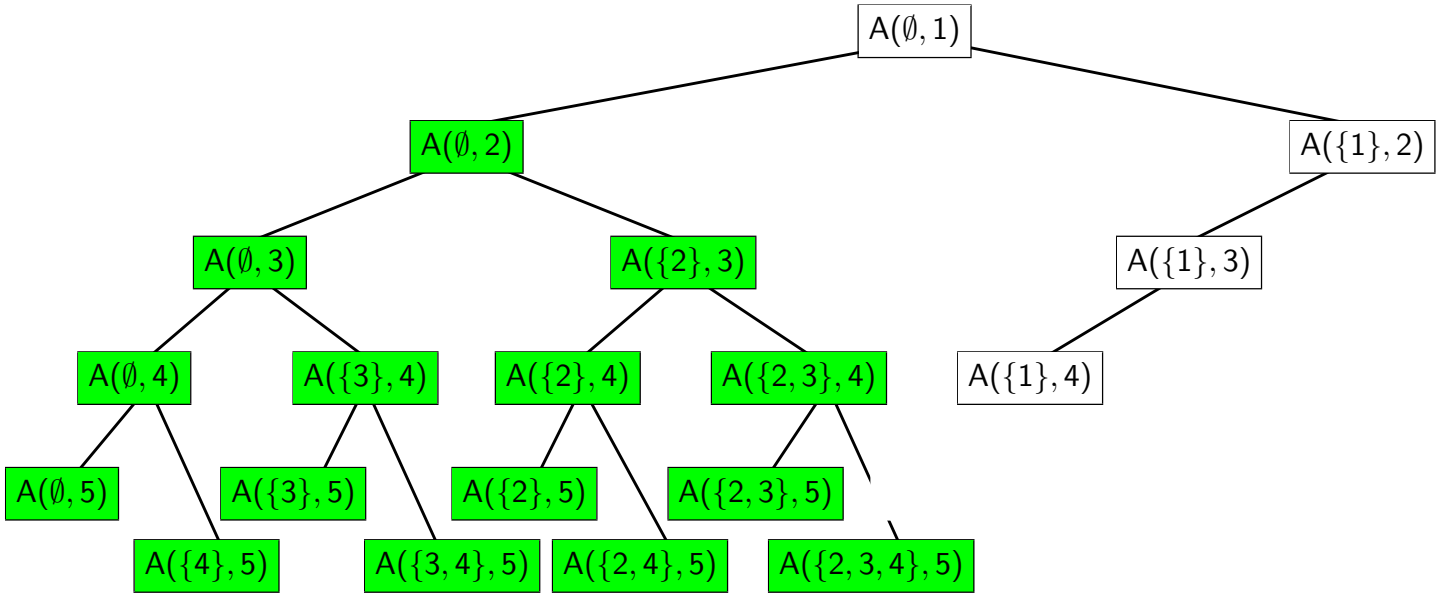


詳細版の動きを見てみる



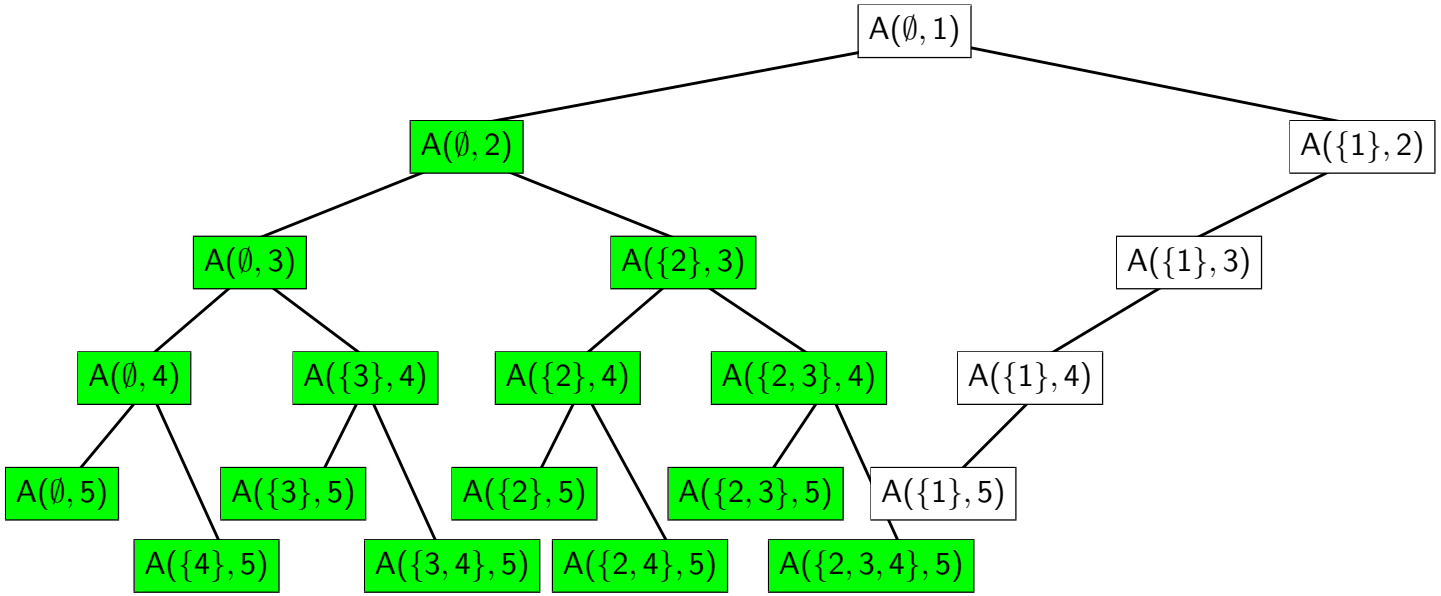
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$

詳細版の動きを見てみる



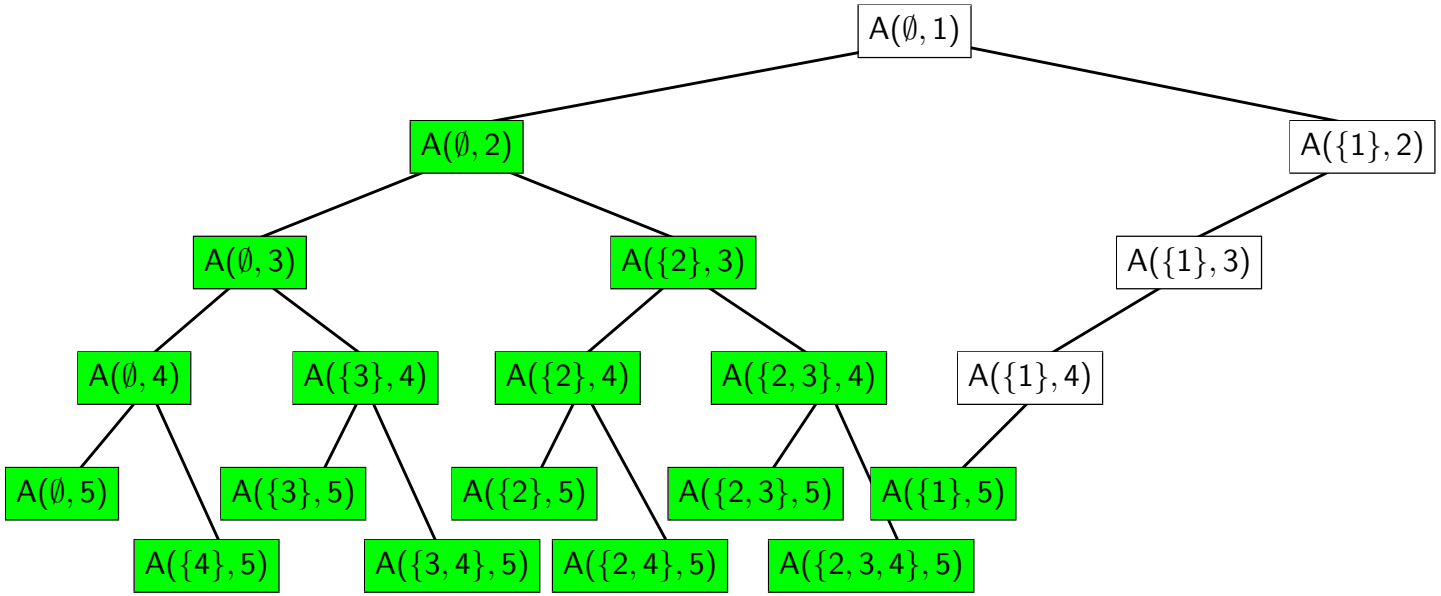
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$

詳細版の動きを見てみる



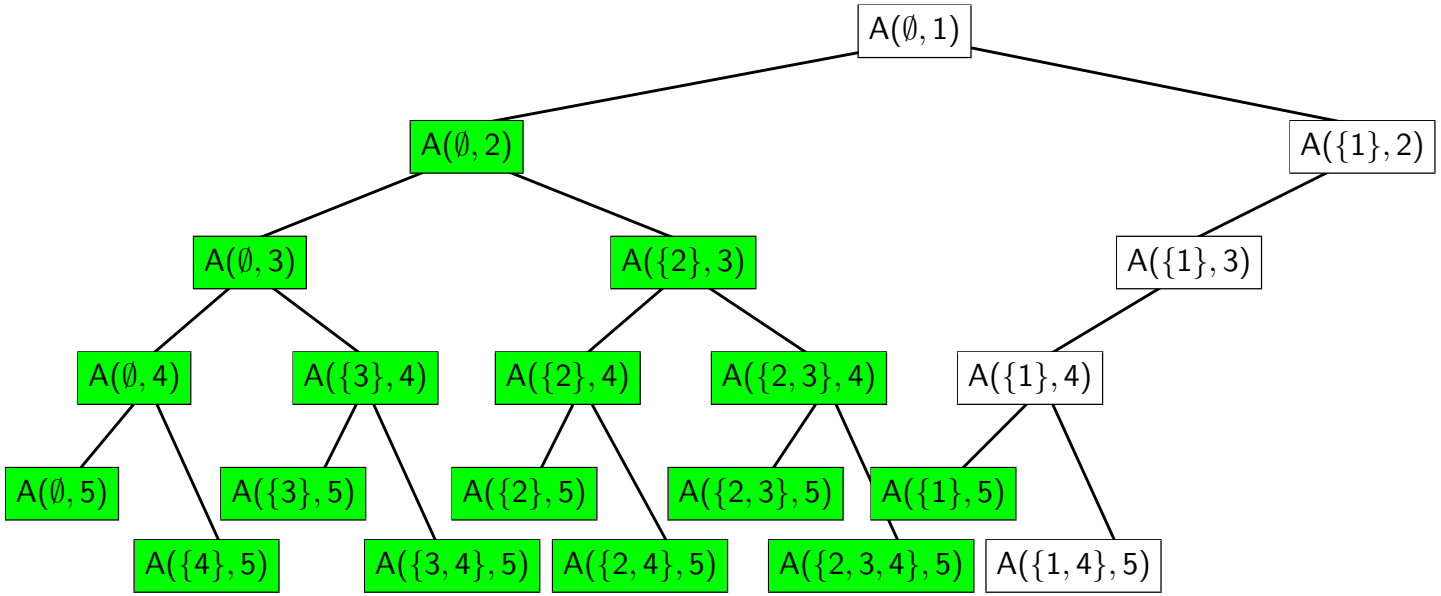
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$

詳細版の動きを見てみる



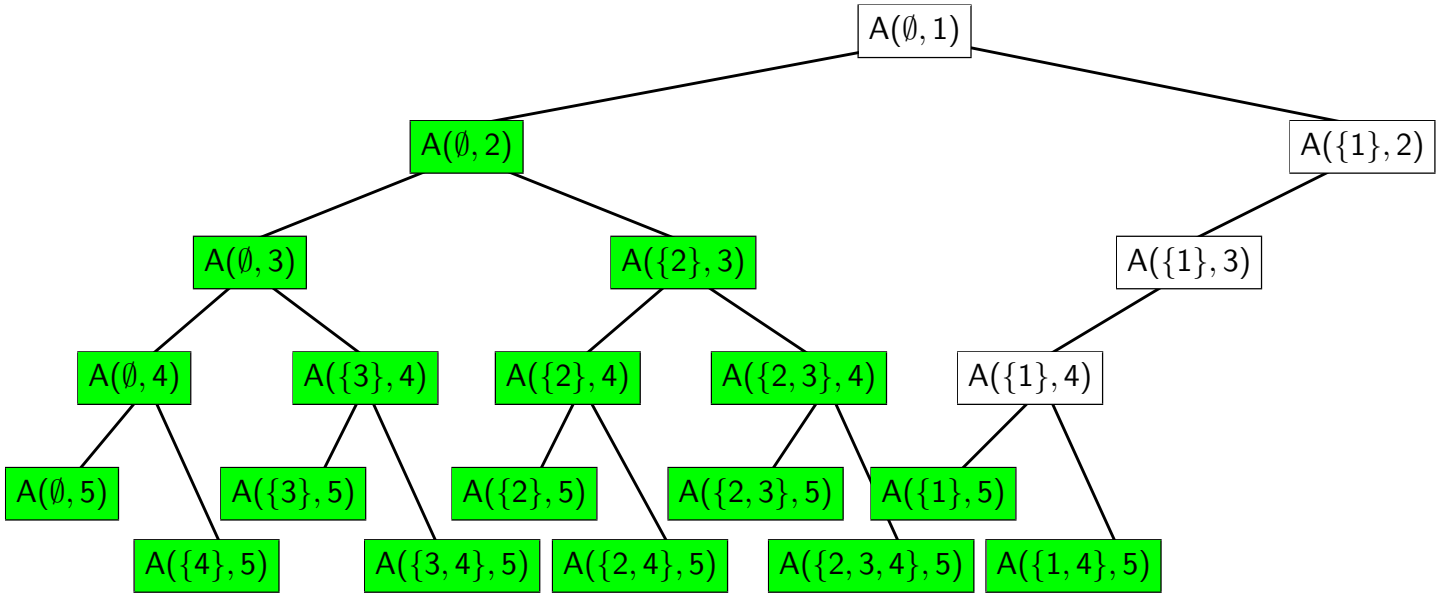
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$

詳細版の動きを見てみる



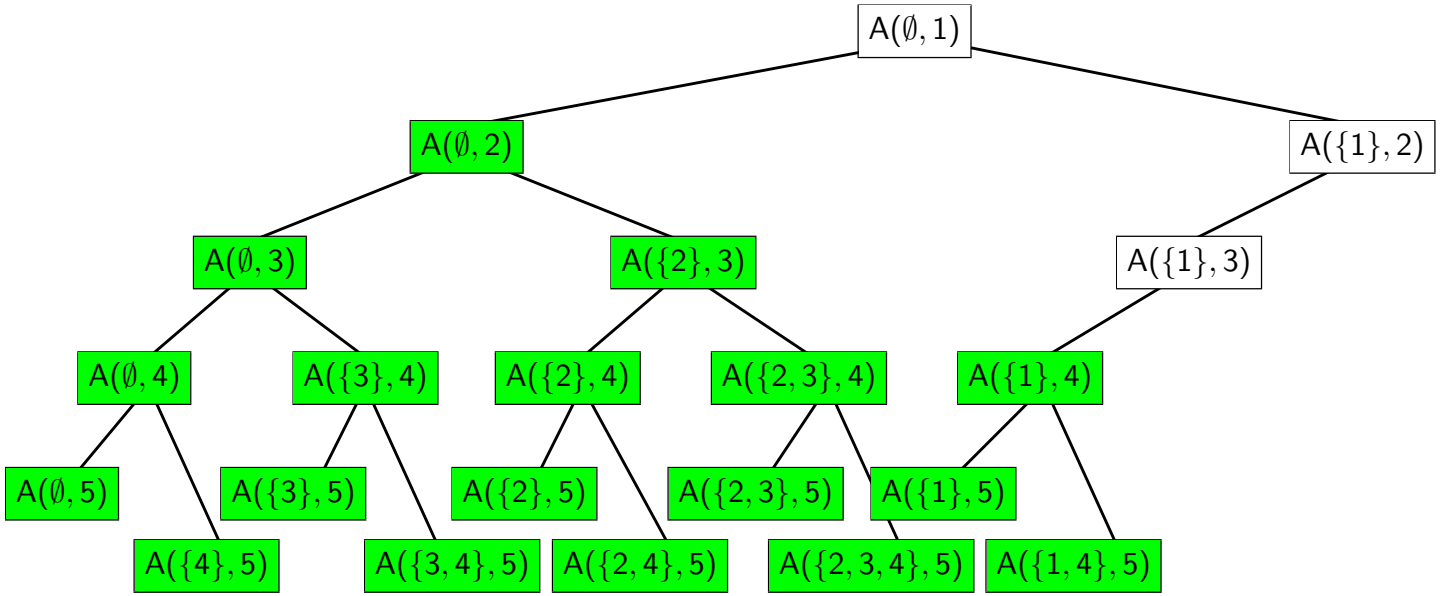
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$

詳細版の動きを見てみる



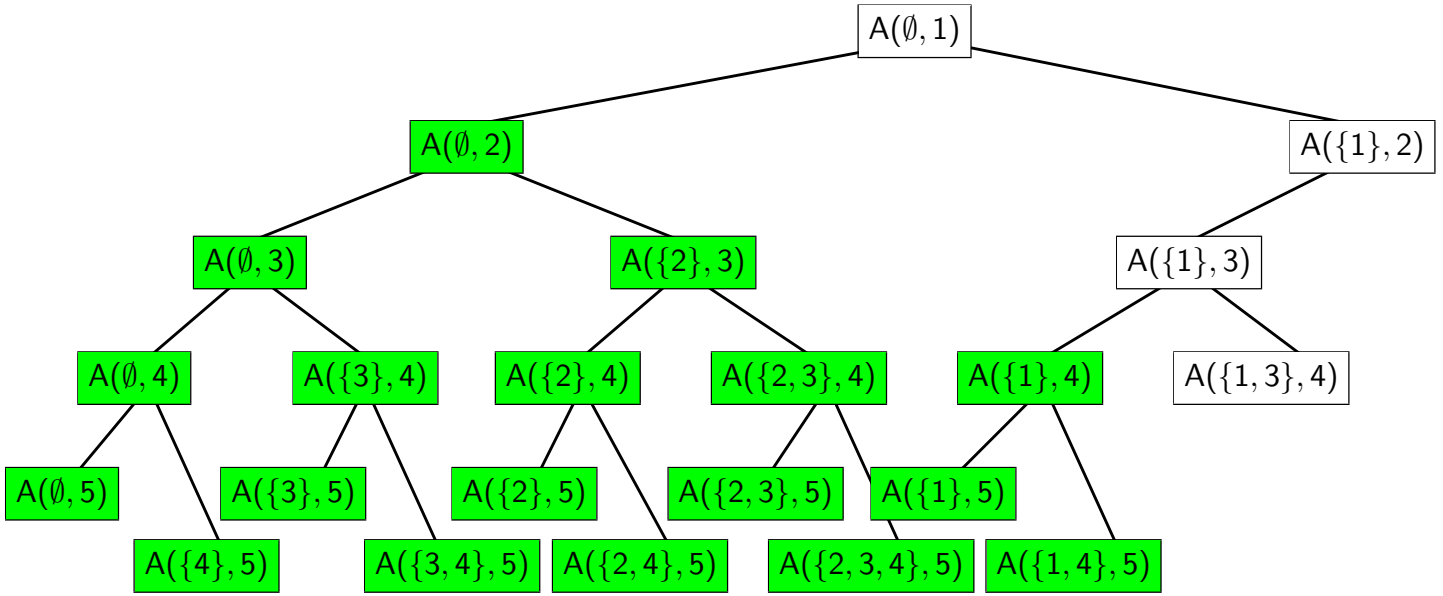
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$

詳細版の動きを見てみる



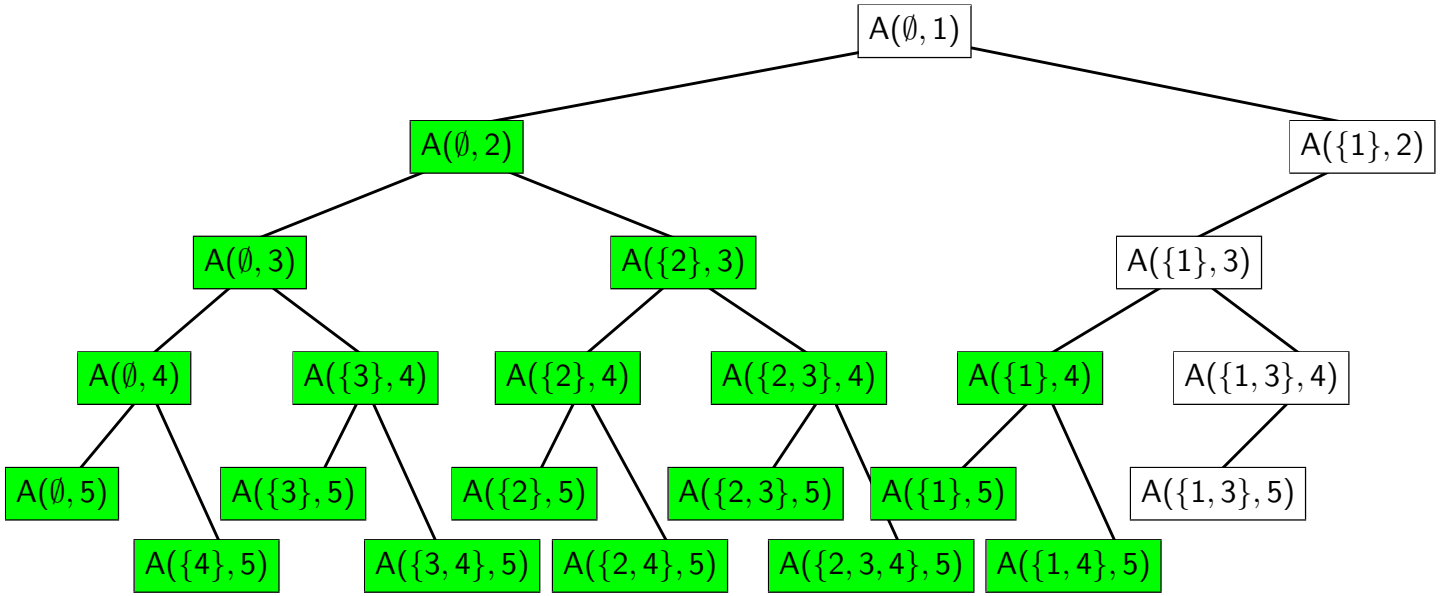
$\emptyset,\ \{4\},\ \{3\},\ \{3,4\},\ \{2\},\ \{2,4\},\ \{2,3\},\ \{2,3,4\},\ \{1\},\ \{1,4\}$

詳細版の動きを見てみる



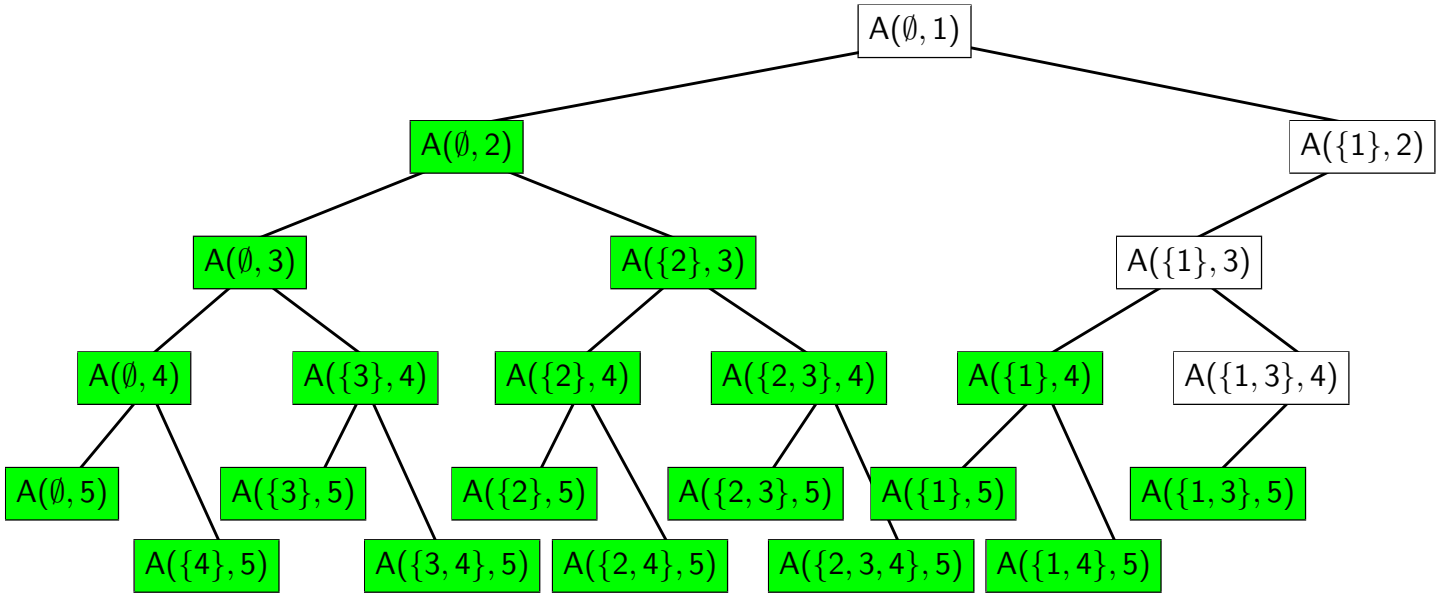
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$

詳細版の動きを見てみる



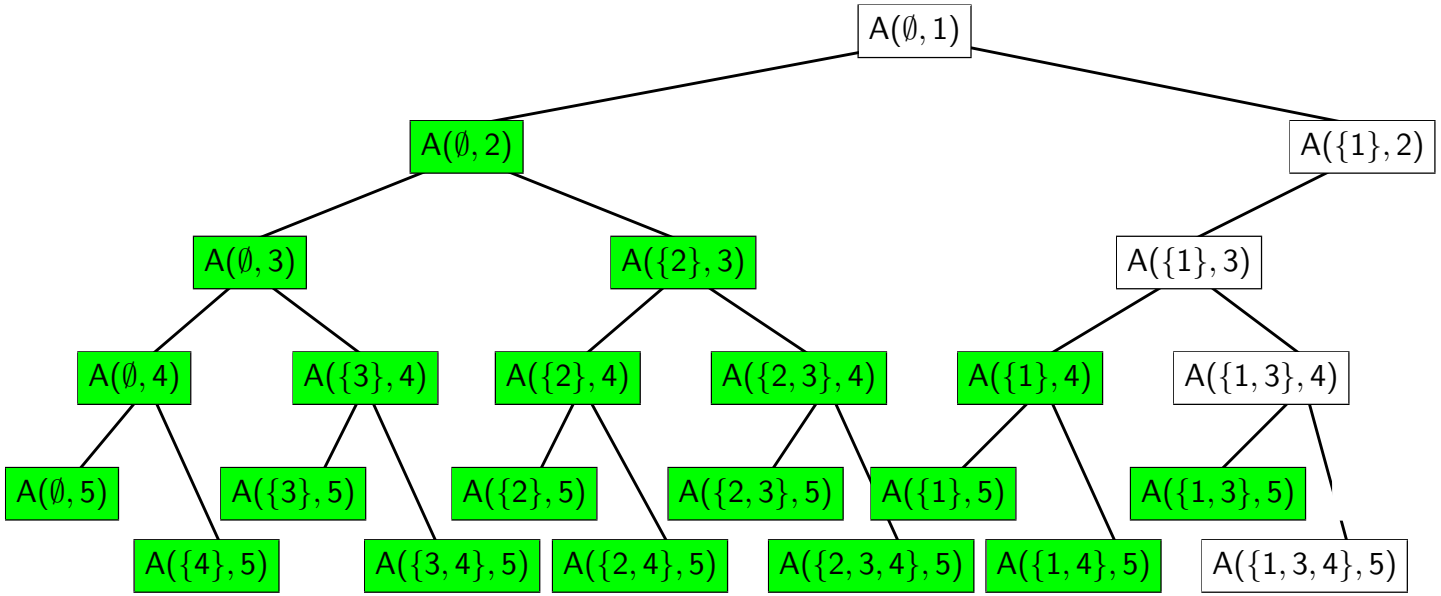
$\emptyset,\ \{4\},\ \{3\},\ \{3,4\},\ \{2\},\ \{2,4\},\ \{2,3\},\ \{2,3,4\},\ \{1\},\ \{1,4\}$

詳細版の動きを見てみる



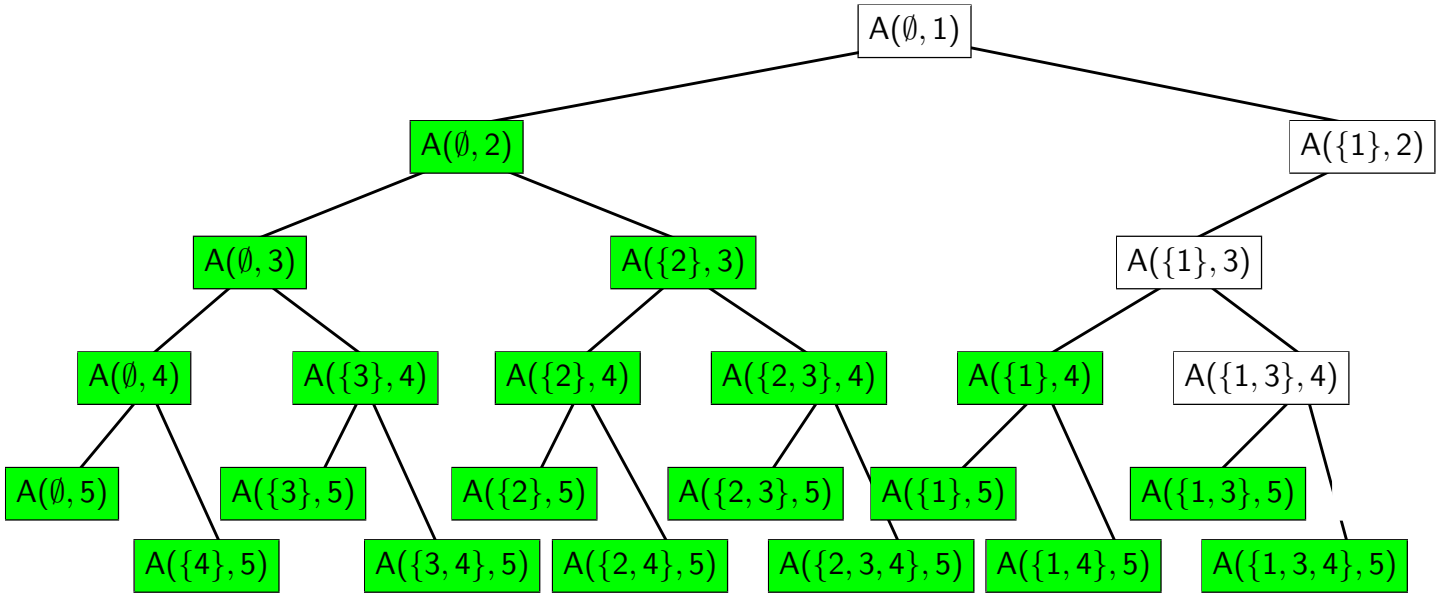
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$

詳細版の動きを見てみる



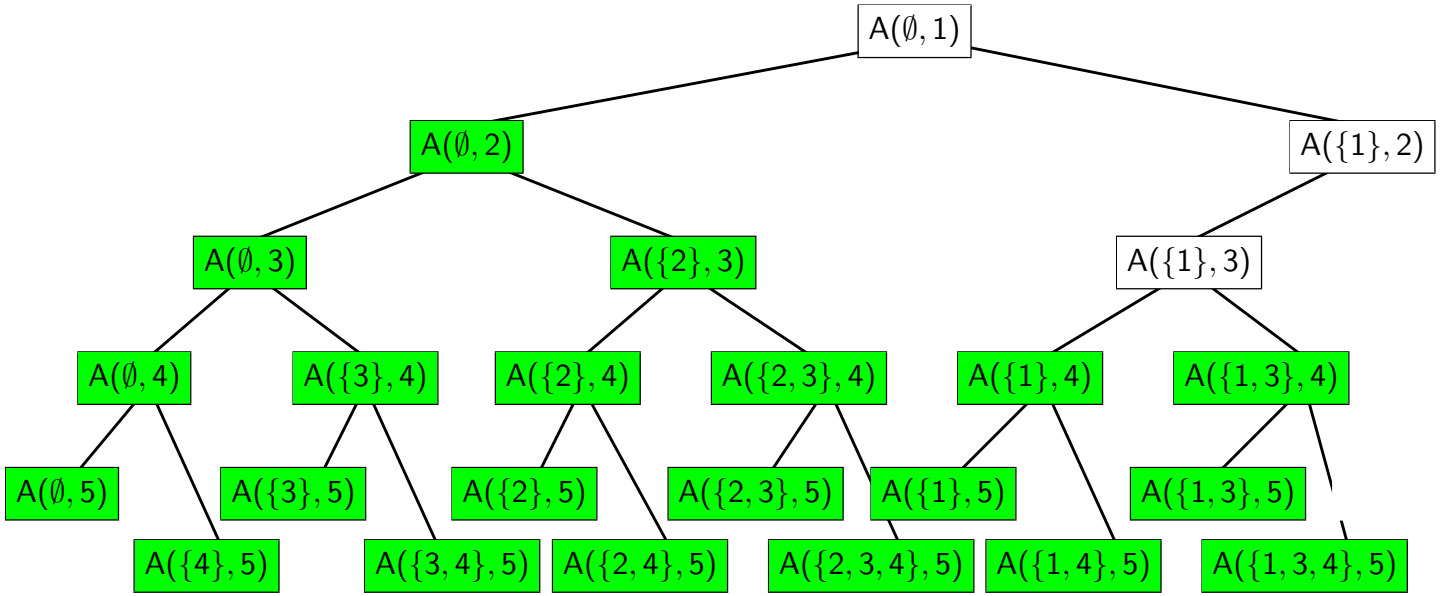
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$

詳細版の動きを見てみる



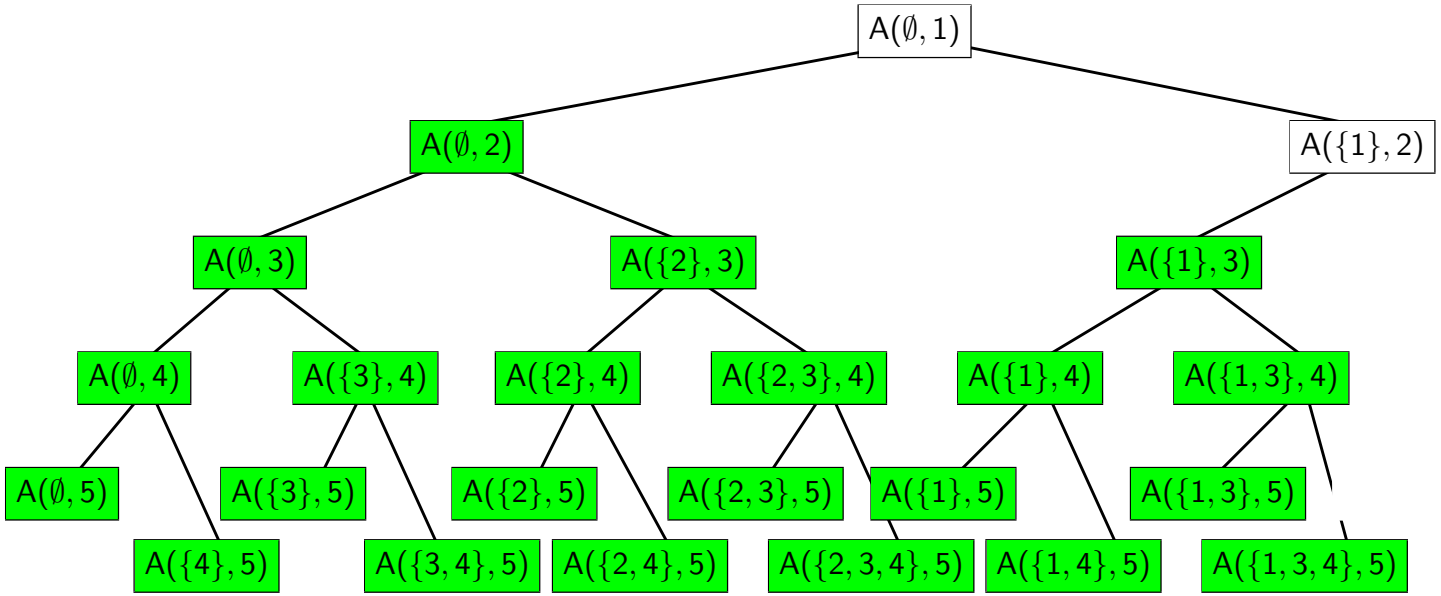
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$

詳細版の動きを見てみる



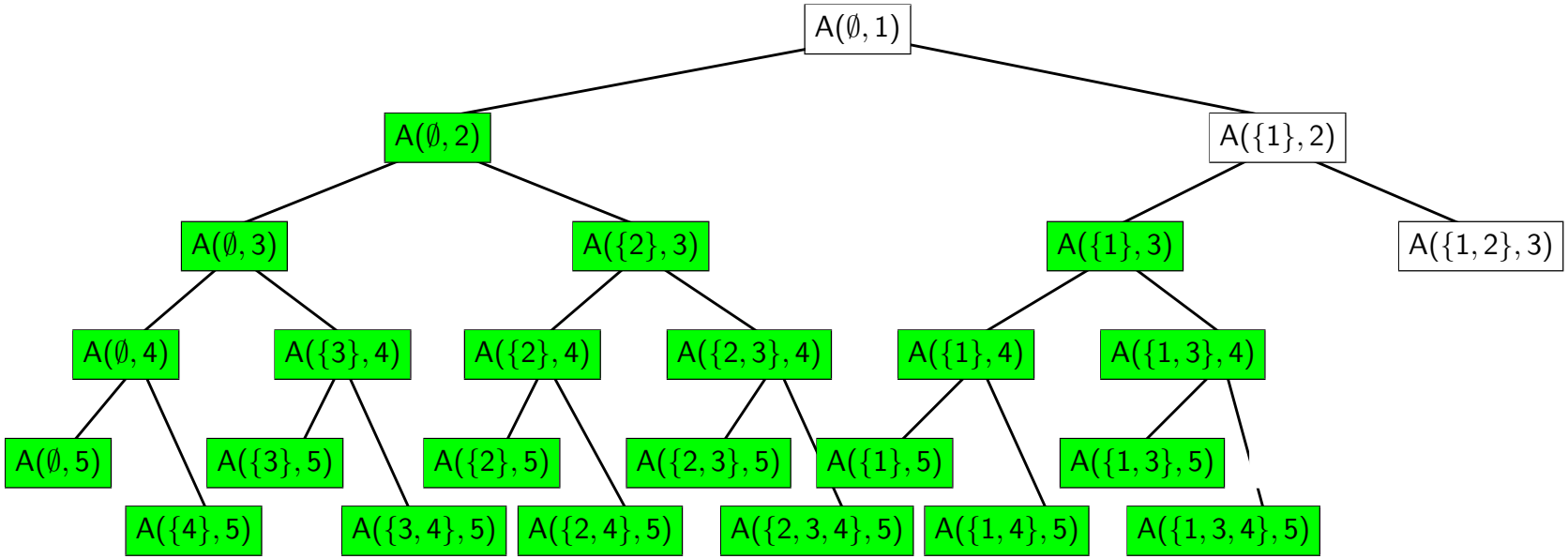
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$

詳細版の動きを見てみる



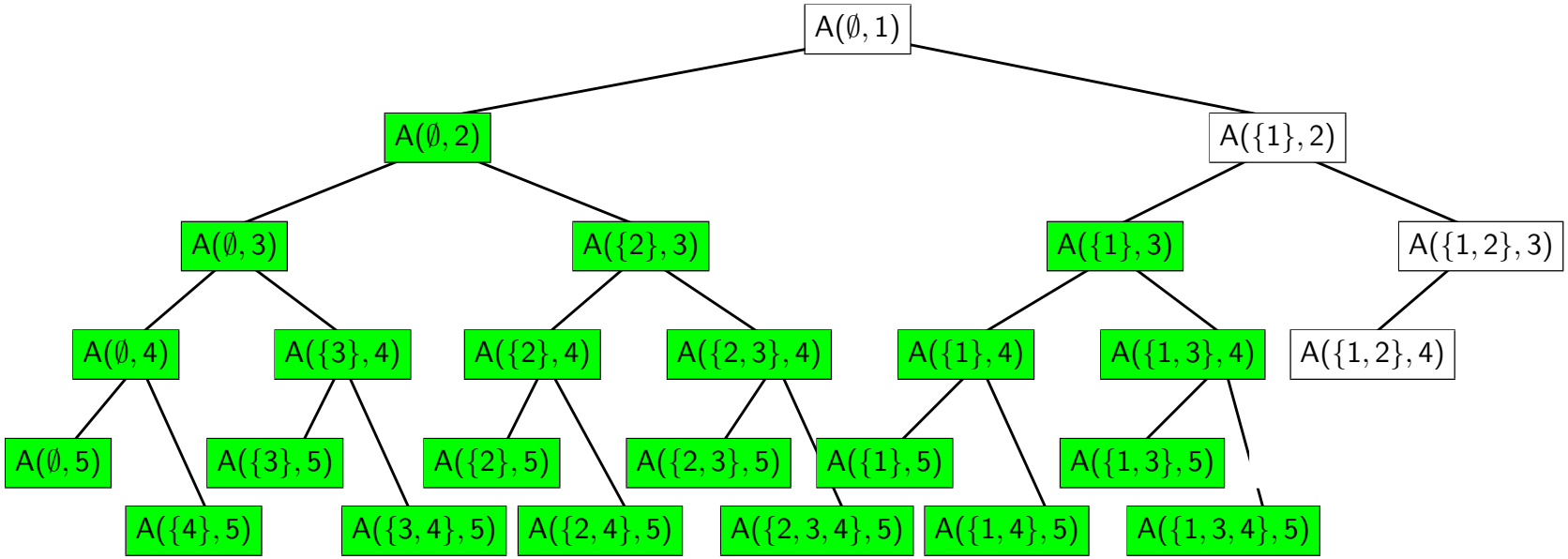
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$

詳細版の動きを見てみる



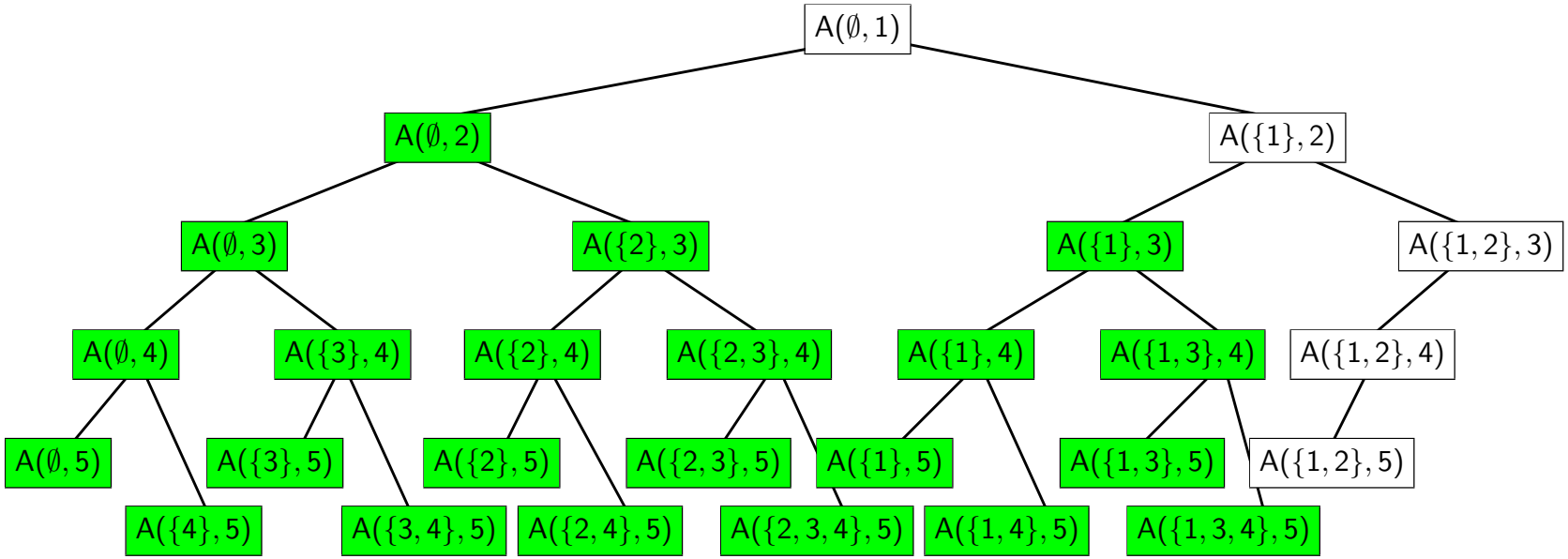
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$

詳細版の動きを見てみる



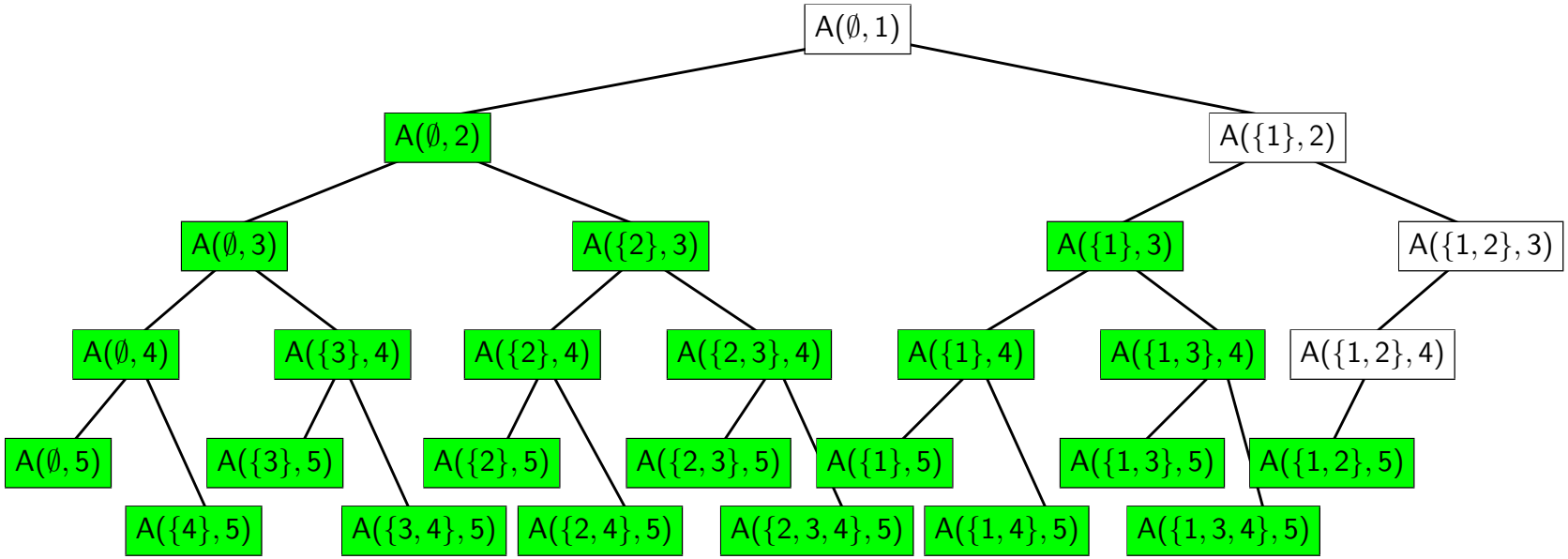
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$

詳細版の動きを見てみる



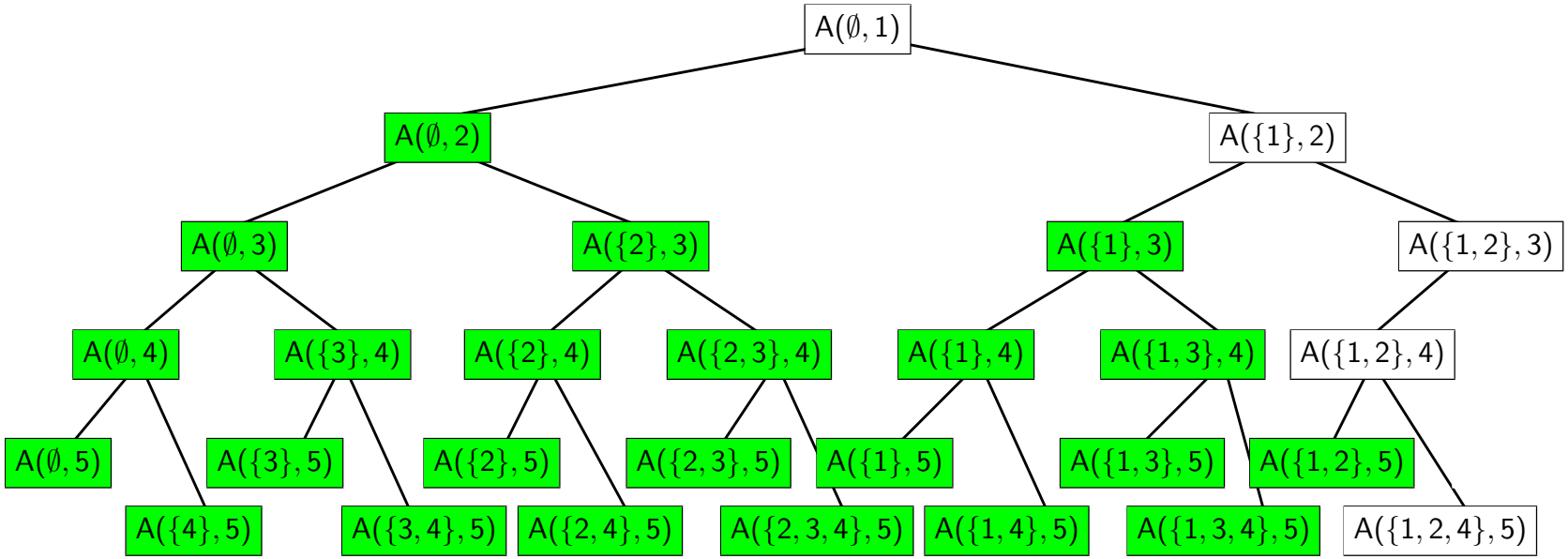
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$

詳細版の動きを見てみる



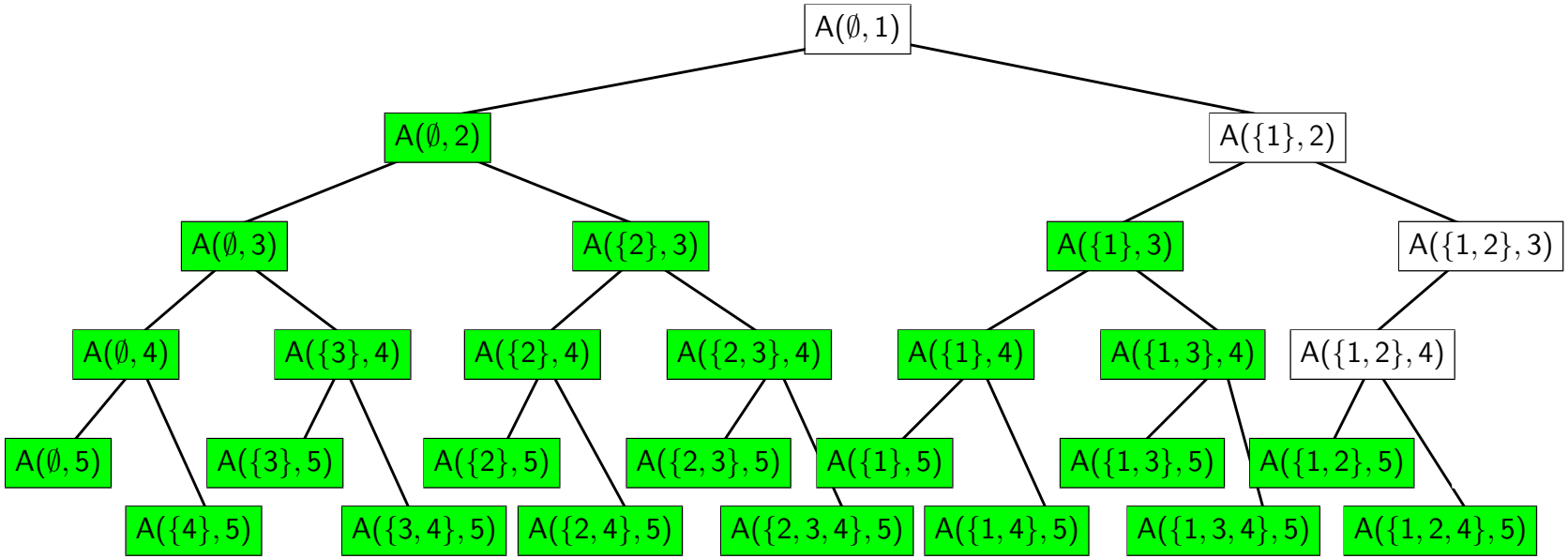
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$

詳細版の動きを見てみる



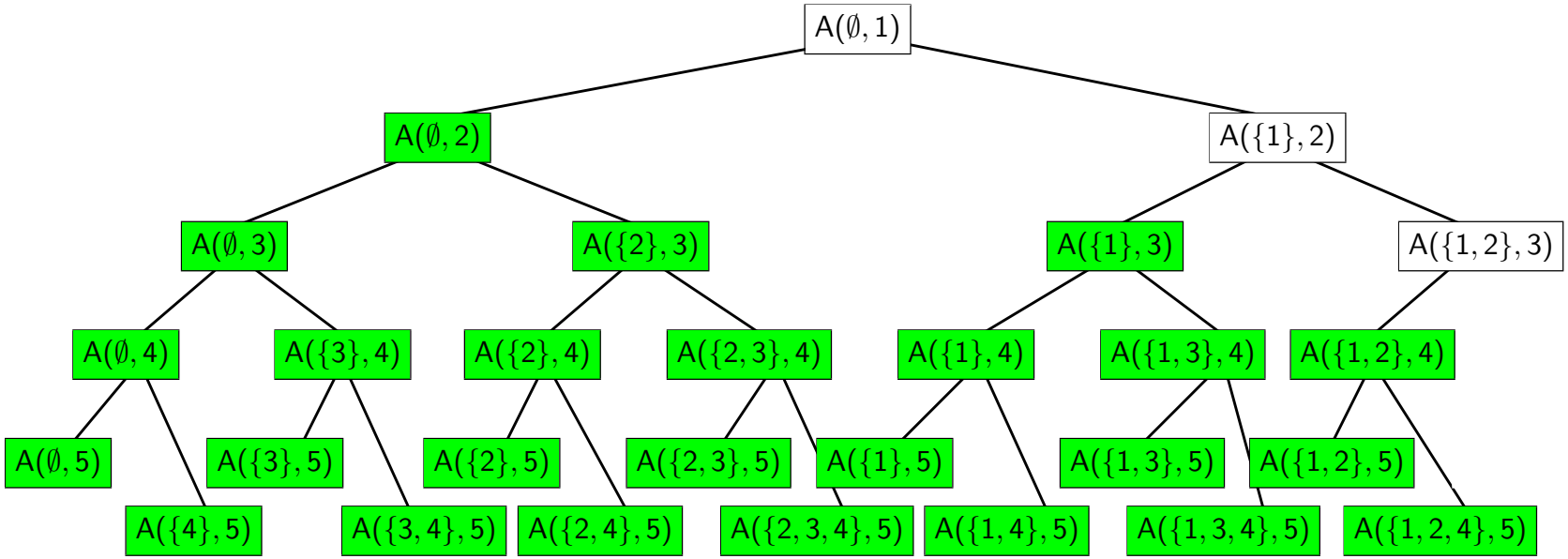
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$

詳細版の動きを見てみる



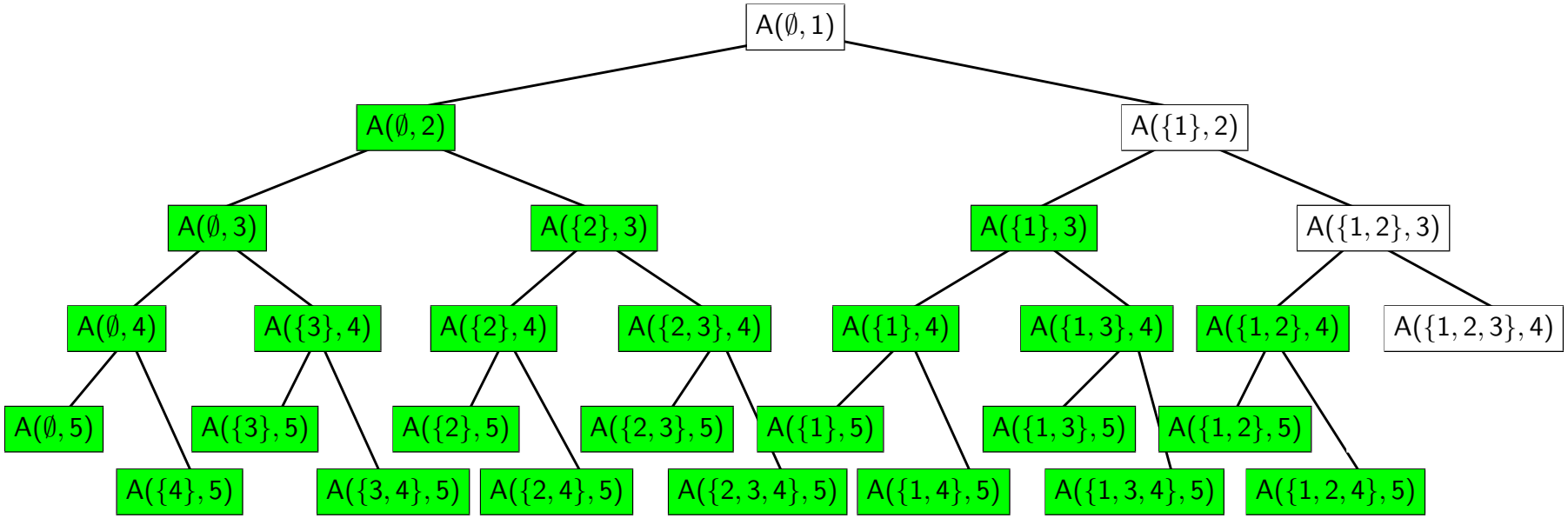
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$

詳細版の動きを見てみる



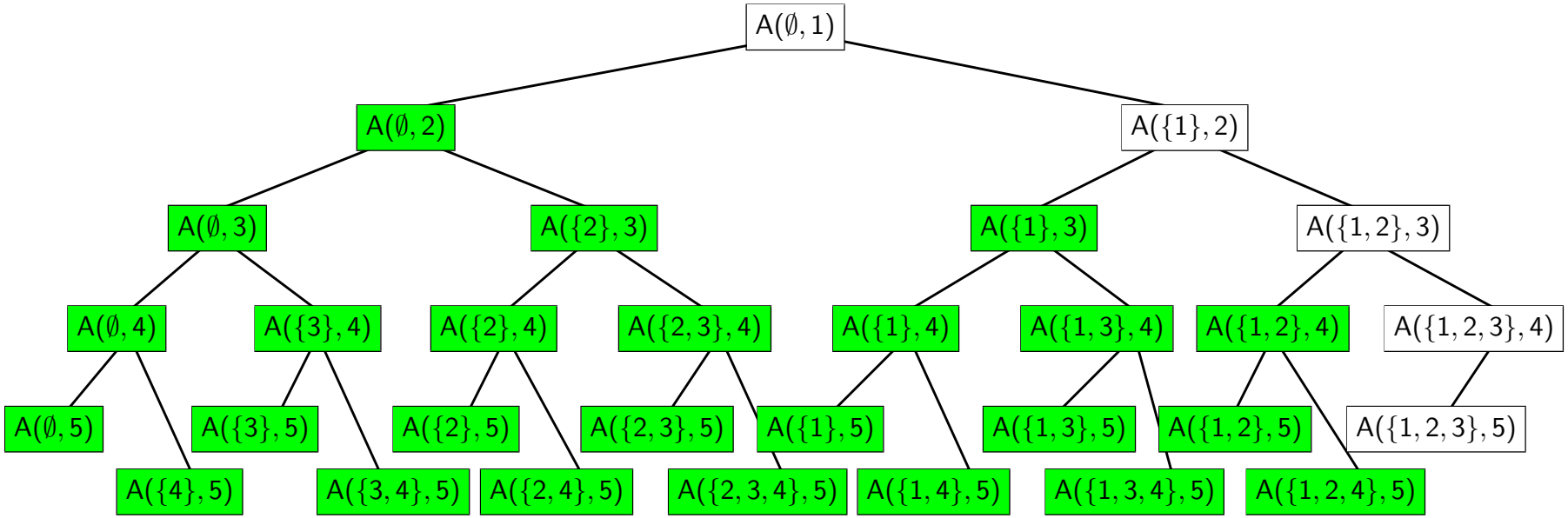
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$

詳細版の動きを見てみる



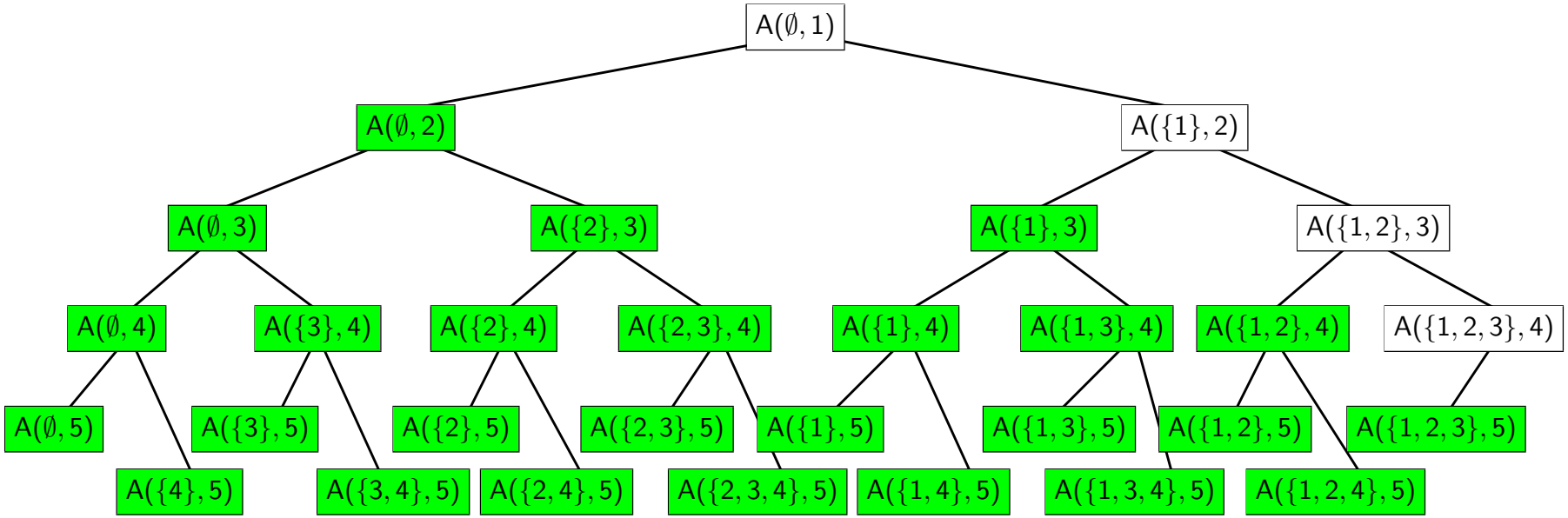
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$

詳細版の動きを見てみる



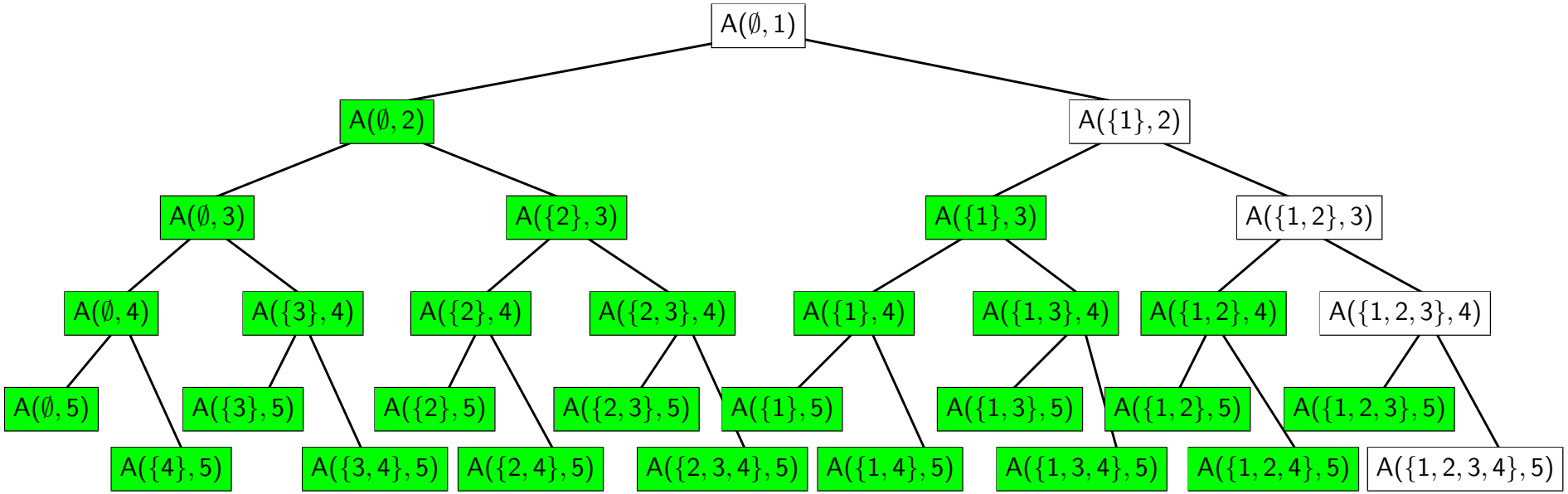
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$

詳細版の動きを見てみる



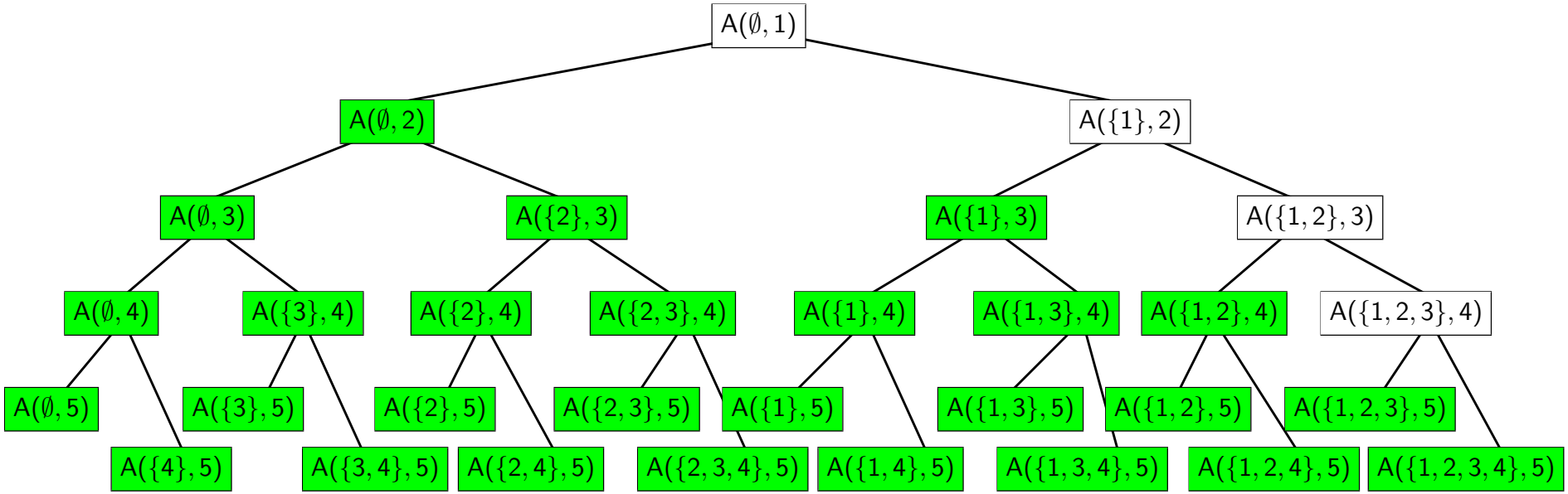
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$, $\{1,2,3\}$

詳細版の動きを見てみる



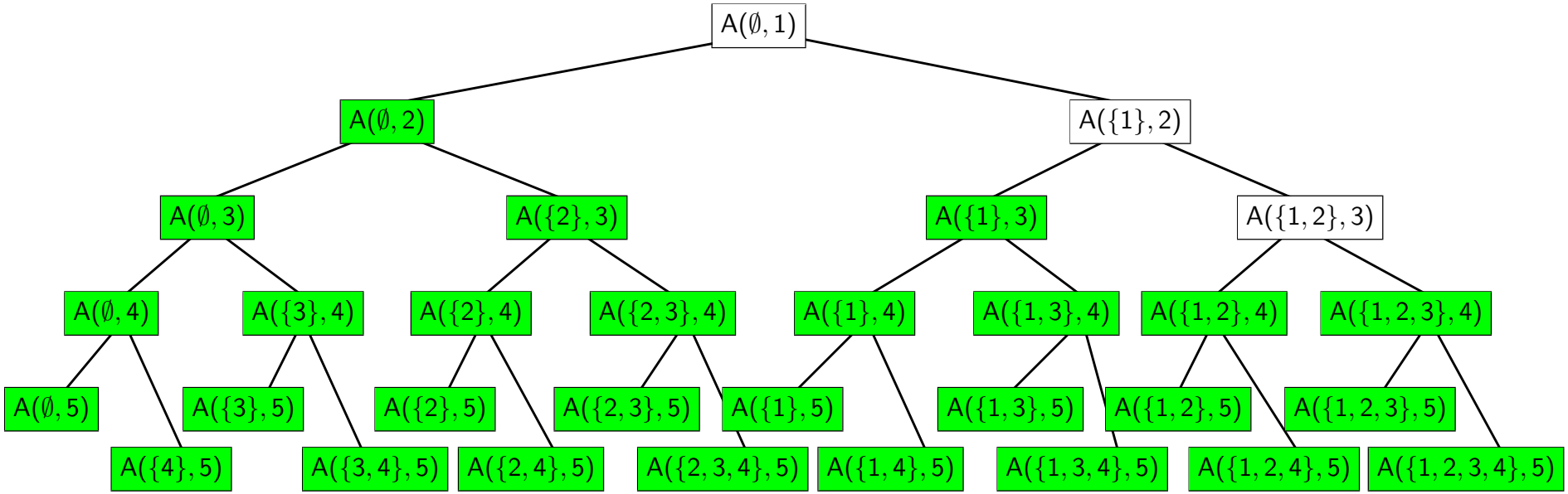
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$, $\{1,2,3\}$

詳細版の動きを見てみる



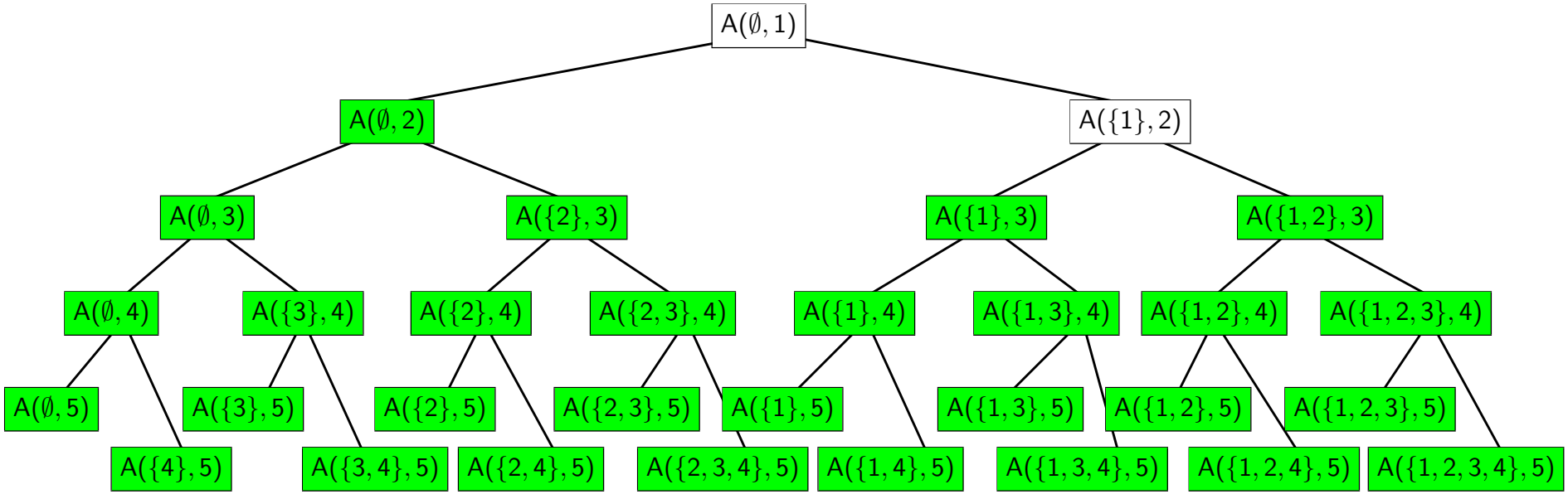
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$, $\{1,2,3\}$, $\{1,2,3,4\}$

詳細版の動きを見てみる



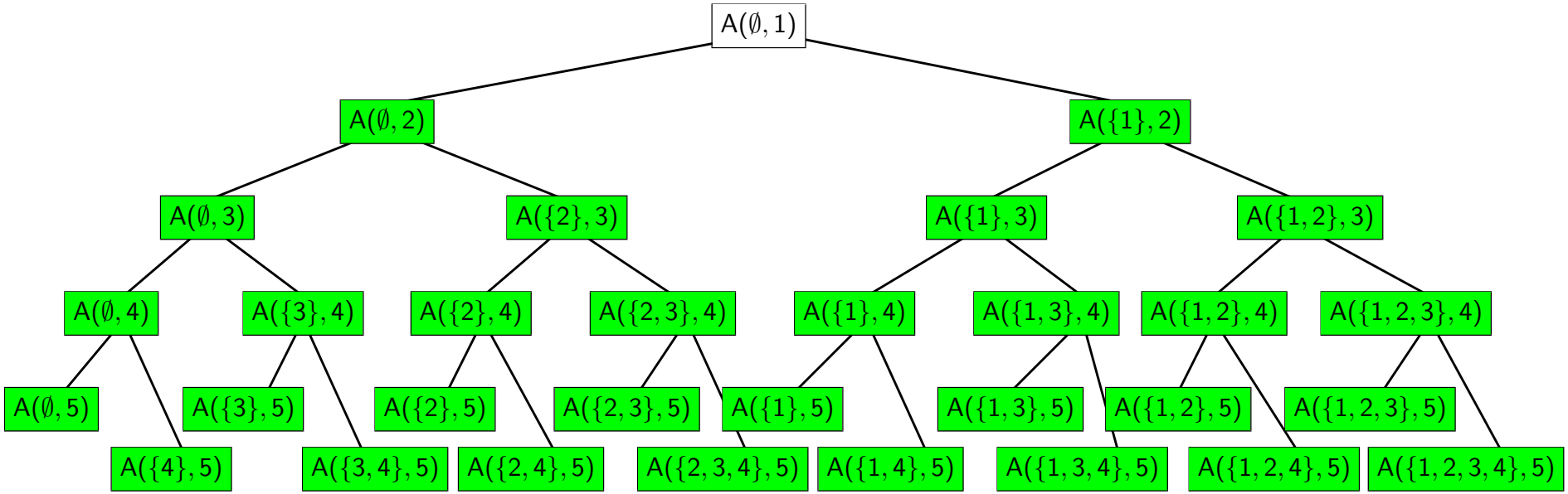
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$, $\{1,2,3\}$, $\{1,2,3,4\}$

詳細版の動きを見てみる



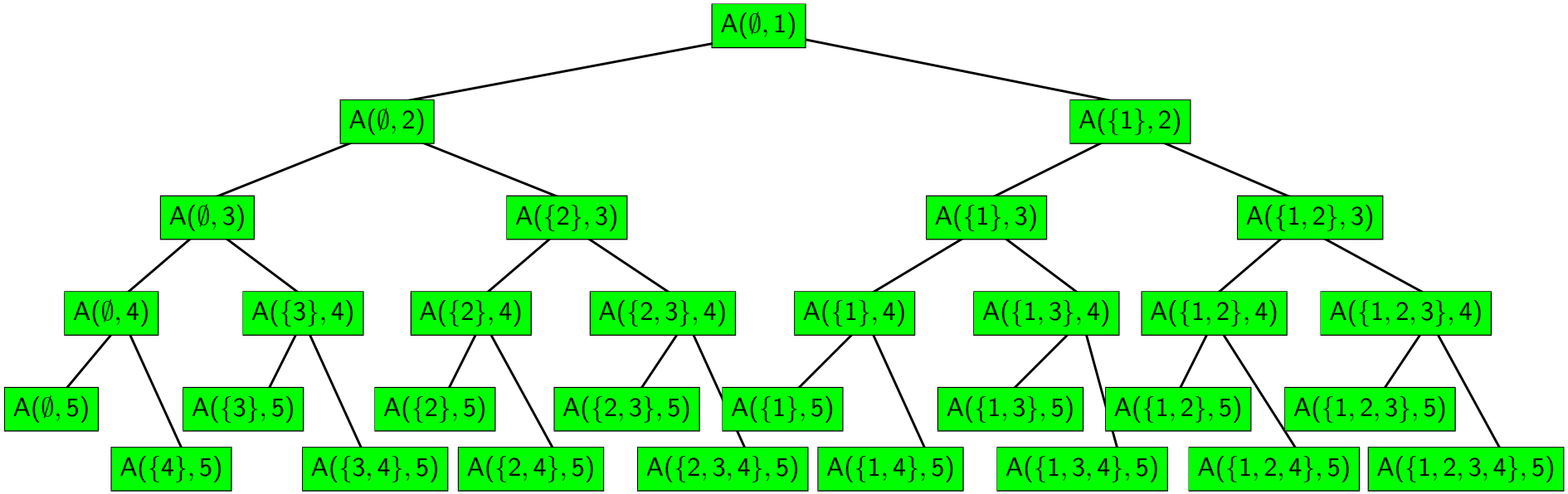
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$, $\{1,2,3\}$, $\{1,2,3,4\}$

詳細版の動きを見てみる



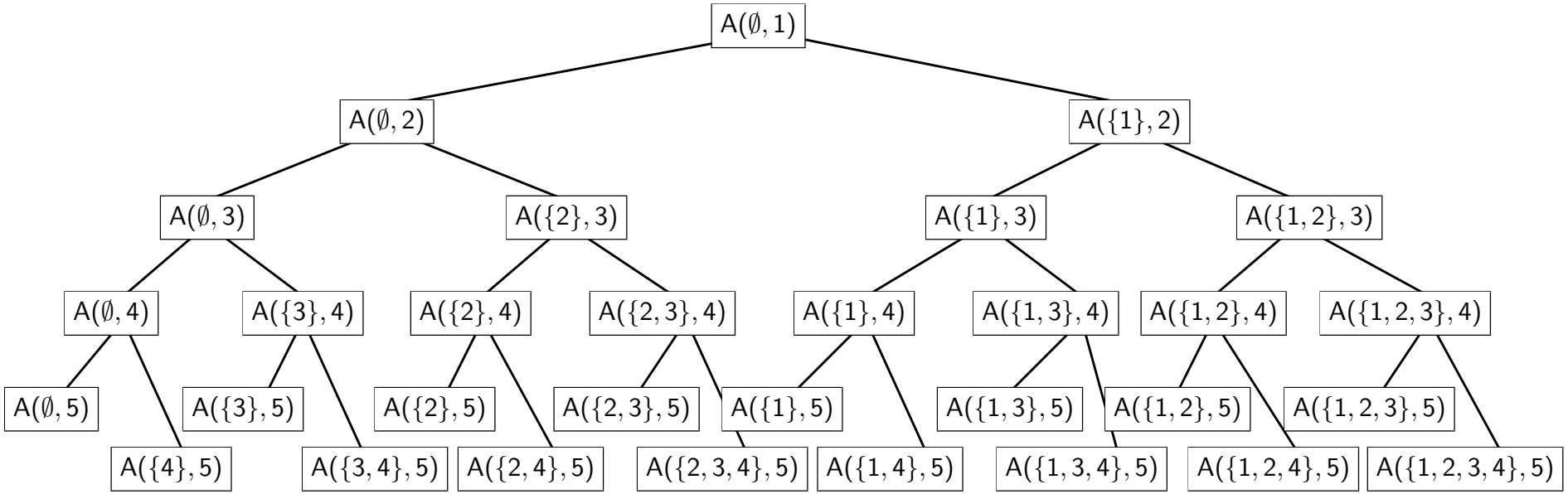
$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$, $\{1,2,3\}$, $\{1,2,3,4\}$

詳細版の動きを見てみる

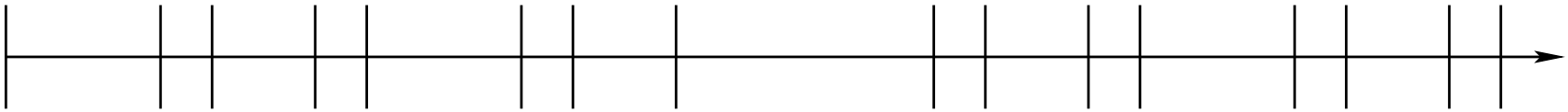


$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$, $\{1,2,3\}$, $\{1,2,3,4\}$

詳細版の動きを見てみた



$\emptyset$, $\{4\}$, $\{3\}$, $\{3,4\}$, $\{2\}$, $\{2,4\}$, $\{2,3\}$, $\{2,3,4\}$, $\{1\}$, $\{1,4\}$, $\{1,3\}$, $\{1,3,4\}$, $\{1,2\}$, $\{1,2,4\}$, $\{1,2,3\}$, $\{1,2,3,4\}$



アルゴリズムの正当性

事前条件を仮定して，事後条件が成り立つことを証明すればよい

$\mathrm{A}(X, i)$

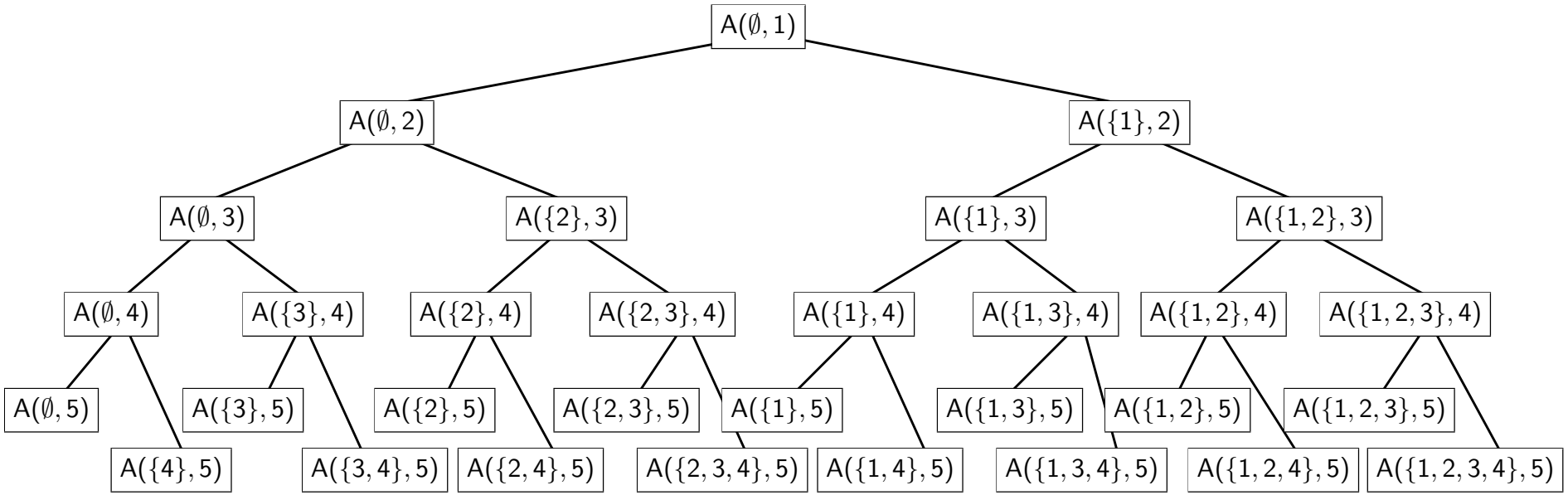
事前条件： $i \in \{1, \ldots, n, n{+}1\}$, $X \subseteq \{1, \ldots, i{-}1\}$
事後条件： $\{X \cup Y \mid Y \subseteq \{i, \ldots, n\}\}$ **の要素をすべて出力**

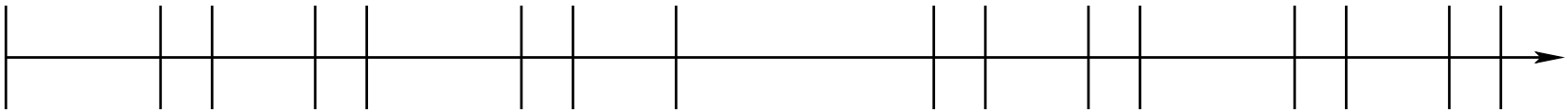
- $i = n{+}1$ **ならば，** X **を出力して終了**
- **そうでなければ，** $\mathrm{A}(X, i{+}1)$ **と** $\mathrm{A}(X \cup \{i\}, i{+}1)$ **を呼び出し**

i に関する帰納法

- $i = n{+}1$ **のときは成立**
- $i \leq n$ **のとき**
  - $\mathrm{A}(X, i{+}1)$ **の出力は** $\{X \cup Y \mid Y \subseteq \{i{+}1, \ldots, n\}\}$
  - $\mathrm{A}(X \cup \{i\}, i{+}1)$ **の出力は** $\{X \cup \{i\} \cup Y \mid Y \subseteq \{i{+}1, \ldots, n\}\}$
  - **これらの合併は** $\{X \cup Y \mid Y \subseteq \{i, \ldots, n\}\}$

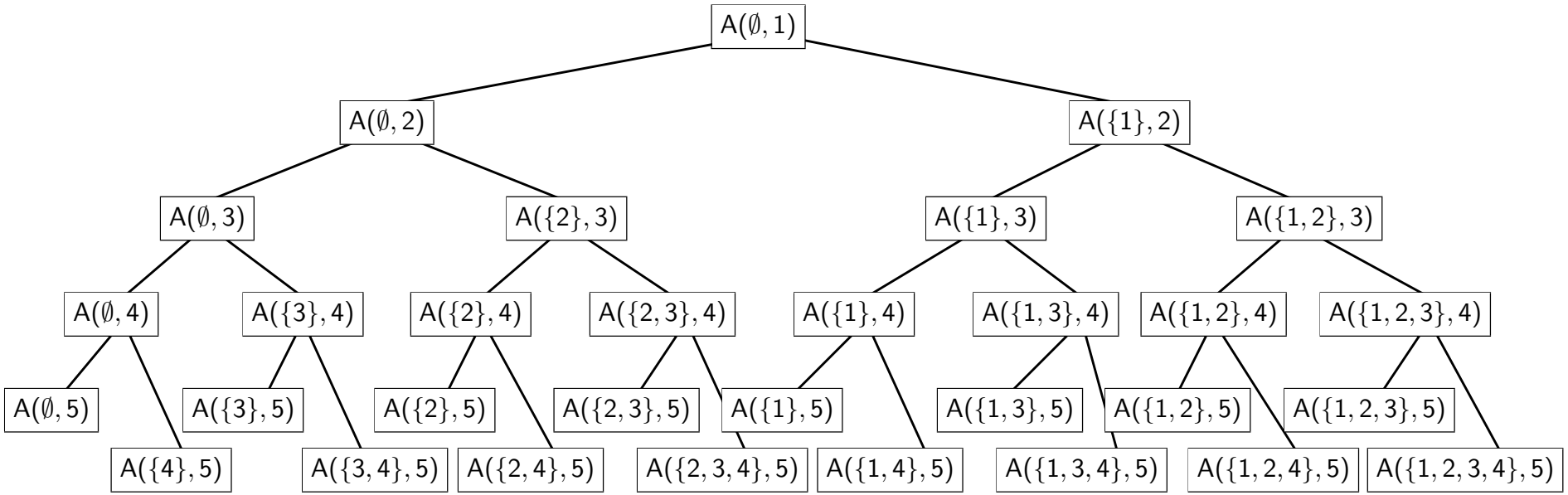
アルゴリズムの効率性 — 時間 (1)

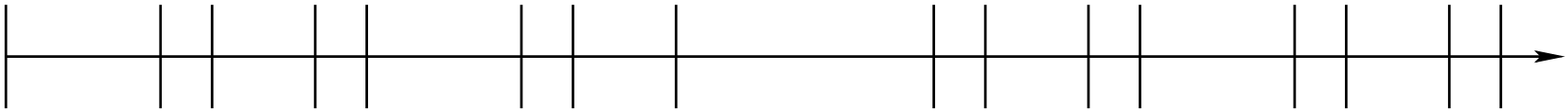




- **時間計算量 $\leq$ 再帰計算木をたどり切る時間**
  $+$ 出力 1 つにかかる最悪時間 $\times$ 出力の数
- $N =$ **出力の数** $(= 2^n)$ **とする**

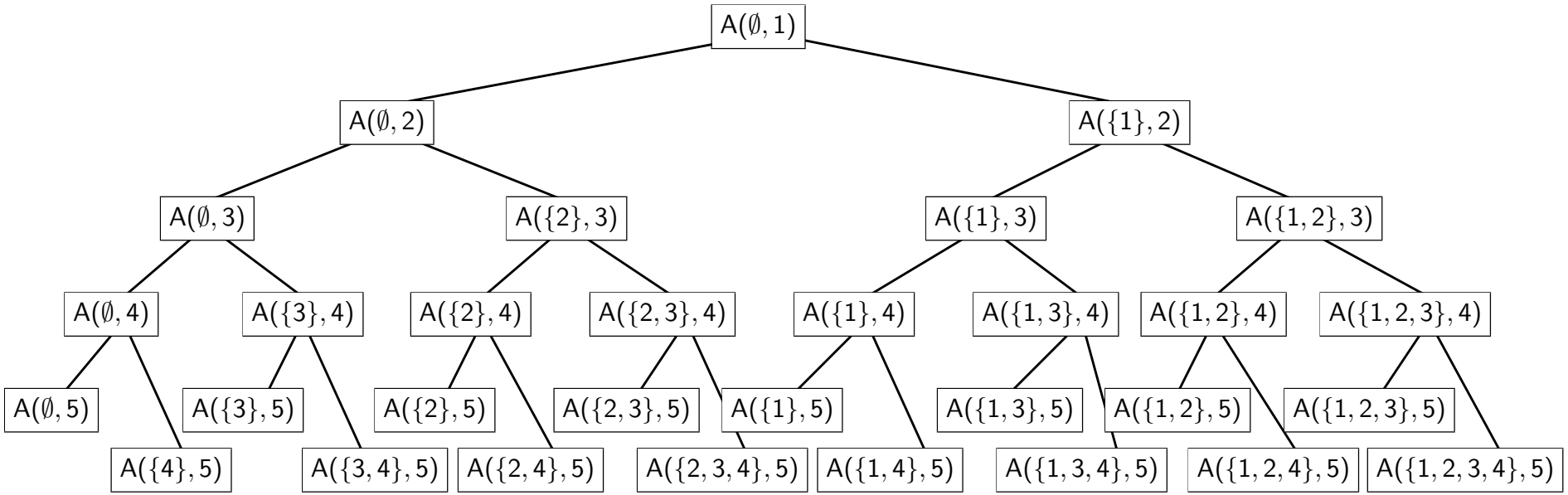
アルゴリズムの効率性 — 時間 (2)

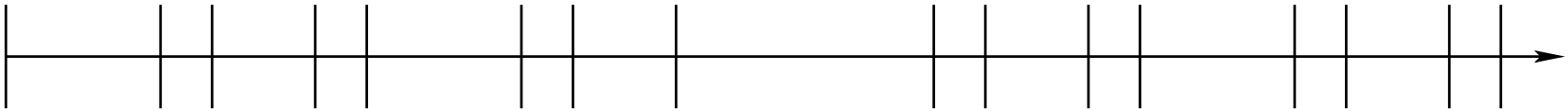




- **再帰計算木の各辺は定数時間でたどれる**
- **再帰計算木の辺数** $= \Theta(N)$
- **出力 1 つにかかる最悪時間** $= \mathrm{O}(n)$
- $\therefore$ **時間計算量** $= \mathrm{O}(N + nN) = \mathrm{O}(nN)$ **(ならし線形時間遅延)**

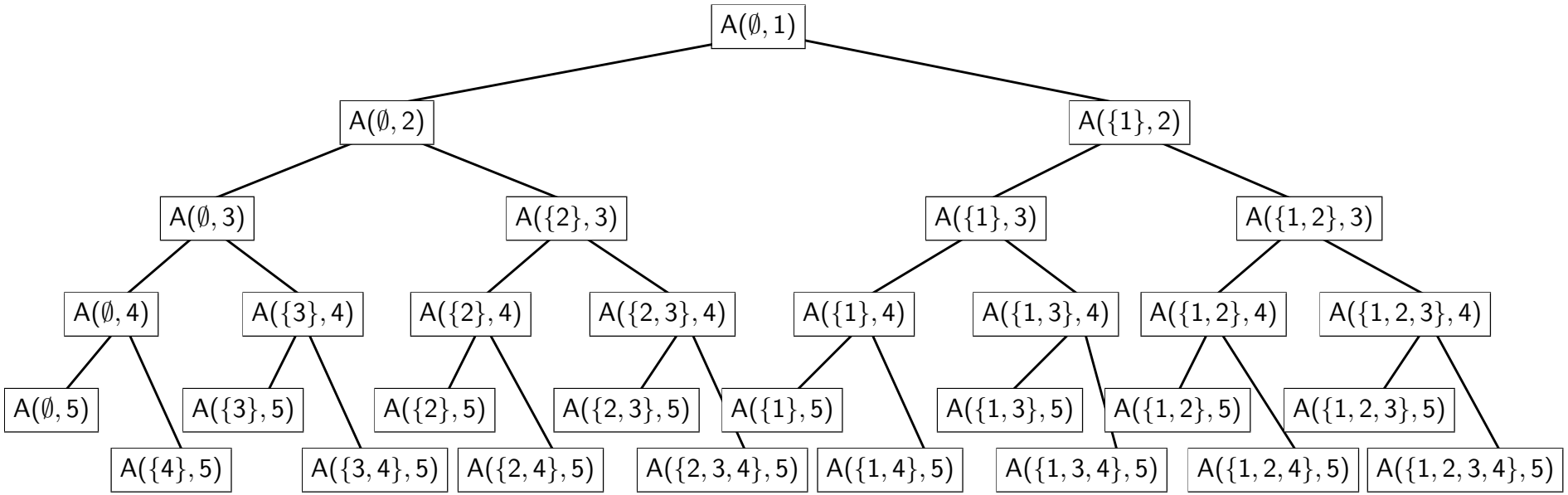
アルゴリズムの効率性 — 領域

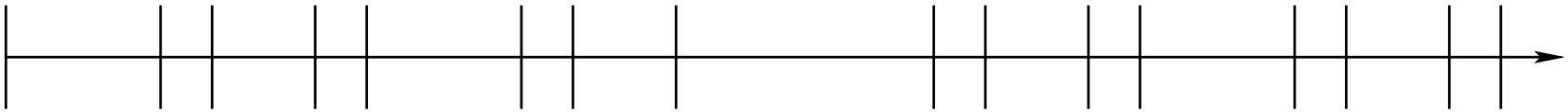




- アルゴリズムの任意の時点で再帰計算木の一部を保存
- 保存している部分の大きさ $= \mathrm{O}(n)$
- 再帰呼び出しする関数の引数の大きさ $= \mathrm{O}(n)$
- $\therefore$ 領域計算量 $= \mathrm{O}(n)$

アルゴリズムの効率性 — 時間 (再)





実は「最悪時線形時間遅延」である

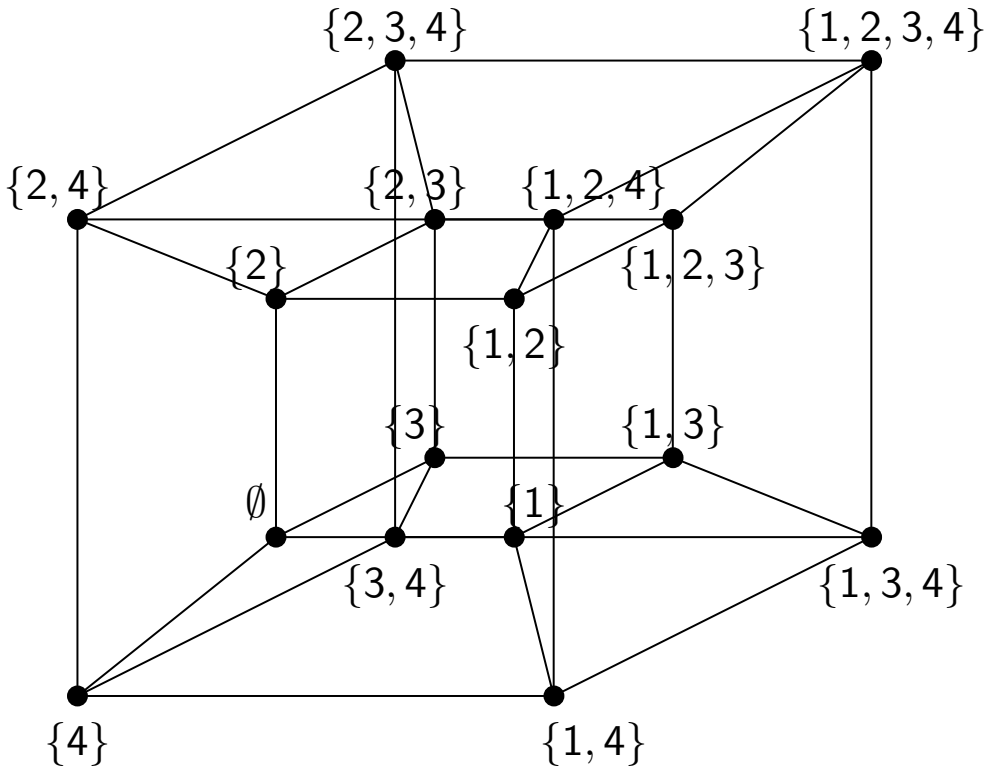
- **左部分木の右端を出力してから右部分木の左端を出力する遅延が最悪時で，これは $\mathrm{O}(n)$**
- **(詳細は略)**

パート 1 の内容

- **列挙問題，列挙アルゴリズムとは?**
- **列挙問題，列挙アルゴリズム設計の難しさ**
- **列挙アルゴリズム設計技法**
  - **分割探索** (binary partition)
  - **組合せ** Gray **符号** (combinatorial Gray code)
  - **逆探索** (reverse search)
- **難しい列挙問題**

基本的な考え方

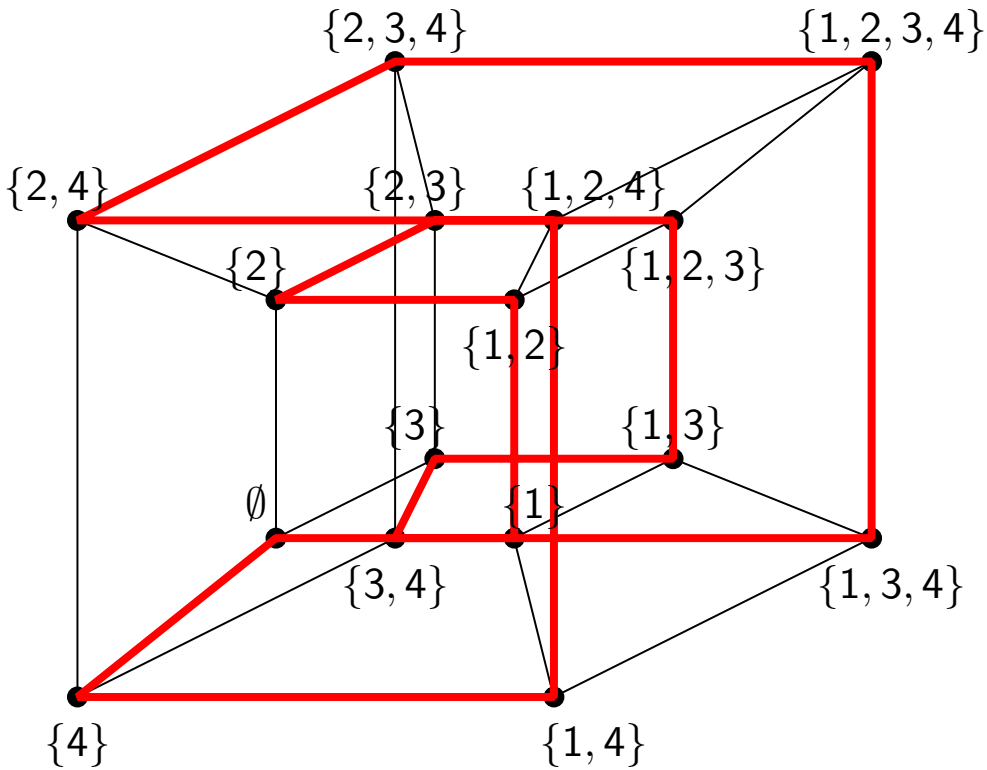
- 部分集合全体の集合の上に無向グラフを定義
- その無向グラフのハミルトン道をたどることで列挙



ハミルトン道 (閉路)：すべての頂点を一度ずつ通過する道 (閉路)

基本的な考え方

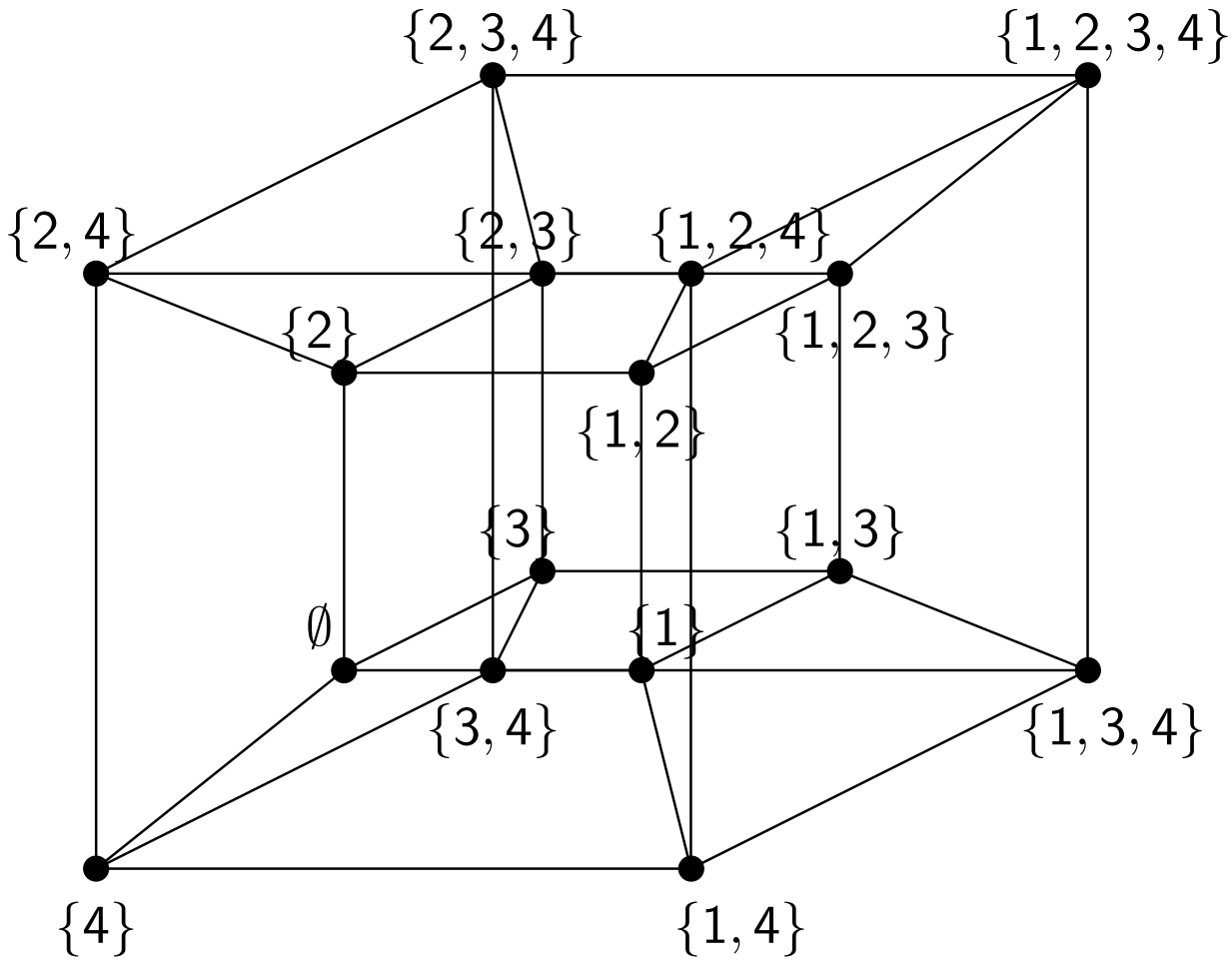
- 部分集合全体の集合の上に無向グラフを定義
- その無向グラフのハミルトン道をたどることで列挙



ハミルトン道 (閉路)：すべての頂点を一度ずつ通過する道 (閉路)

部分集合列挙問題に対するグラフ Q_n

n 次元 Hamming 立方体とも呼ばれる



$X, Y \subseteq \{1, \ldots, n\}$ **が隣接** $\iff |X \triangle Y| = 1$ **(対称差の要素数が** 1**)**

Q_n のハミルトン性

命題 1

任意の $n \geq 1$ に対し，Q_n は $\emptyset$ から $\{n\}$ へのハミルトン道を持つ

証明：n に関する帰納法

- **$n = 1$ のとき成立**

Q_n のハミルトン性

命題 1

任意の $n \geq 1$ に対し，Q_n は $\emptyset$ から $\{n\}$ へのハミルトン道を持つ

証明：n に関する帰納法

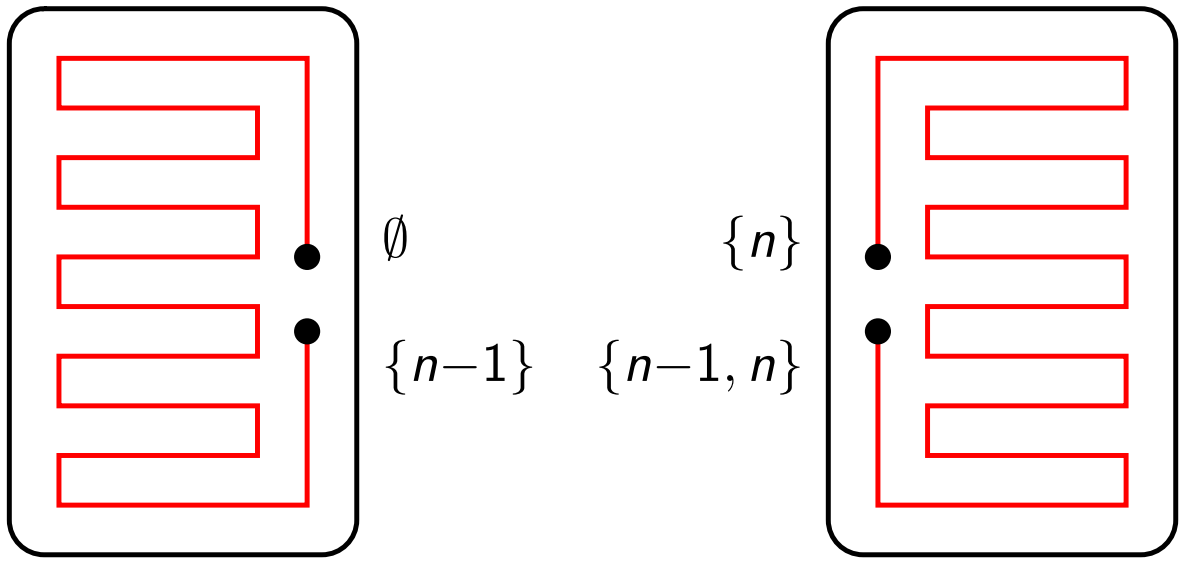
- **$n = 1$ のとき成立**
- **$n > 1$ のとき，**

命題 1

任意の $n \geq 1$ に対し，Q_n は $\emptyset$ から $\{n\}$ へのハミルトン道を持つ

証明：n に関する帰納法

- $n = 1$ のとき成立
- $n > 1$ のとき，
  - n を含む集合の誘導する部分グラフ $\simeq Q_{n-1}$
  - n を含まない集合の誘導する部分グラフ $\simeq Q_{n-1}$

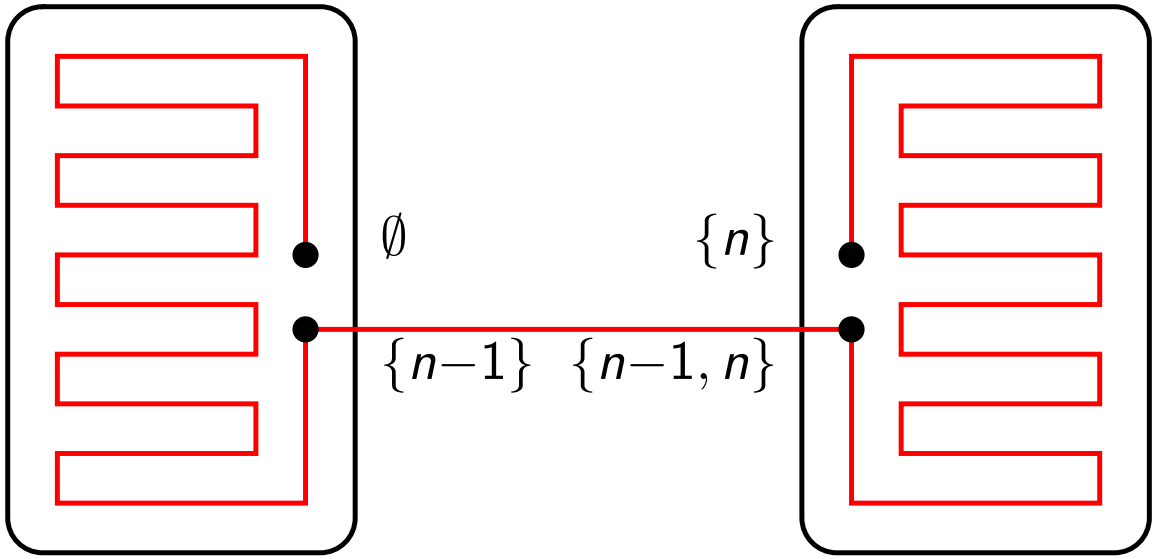


命題 1

任意の $n \geq 1$ に対し，Q_n は $\emptyset$ から $\{n\}$ へのハミルトン道を持つ

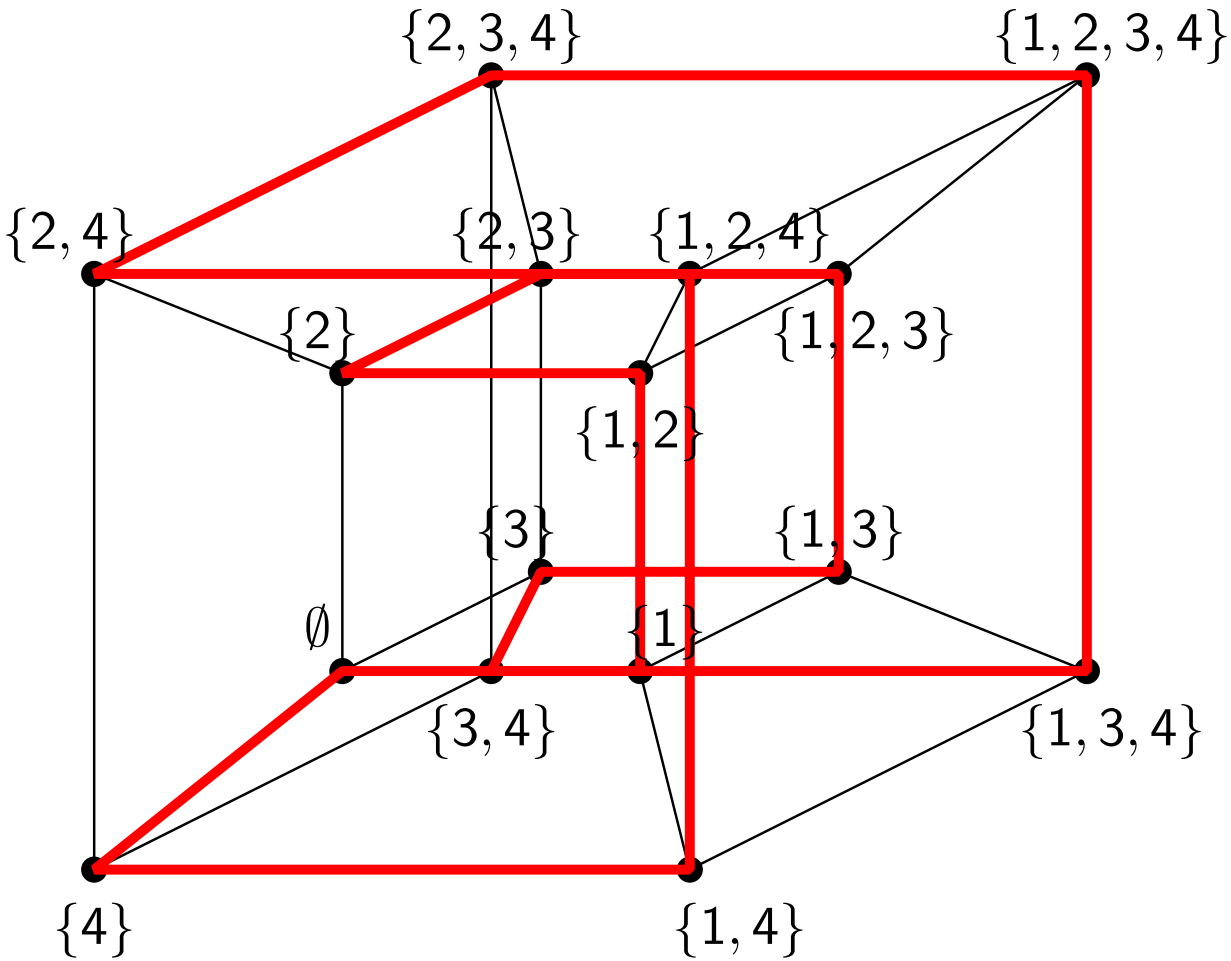
証明：n に関する帰納法

- $n = 1$ のとき成立
- $n > 1$ のとき，
  - n を含む集合の誘導する部分グラフ $\simeq Q_{n-1}$
  - n を含まない集合の誘導する部分グラフ $\simeq Q_{n-1}$



- $\{n-1\}$ と $\{n-1, n\}$ をつないで，ハミルトン道を得る

ハミルトン道に沿った列挙



$\emptyset, \{1\}, \{1,2\}, \{2\}, \{2,3\}, \{1,2,3\}, \{1,3\}, \{3\},$
$\{3,4\}, \{1,3,4\}, \{1,2,3,4\}, \{2,3,4\}, \{2,4\}, \{1,2,4\}, \{1,4\}, \{4\}$

アルゴリズム設計のために理解したいこと

集合 X を出力した後，次に出力される集合 X' をすぐに見つける方法

$\emptyset, \{1\}, \{1,2\}, \{2\}, \{2,3\}, \{1,2,3\}, \{1,3\}, \{3\},$
$\{3,4\}, \{1,3,4\}, \{1,2,3,4\}, \{2,3,4\}, \{2,4\}, \{1,2,4\}, \{1,4\}, \{4\}$

命題 2

命題 1 の証明で構成した Q_n のハミルトン道を
$\emptyset, \{1\}, \ldots$ と順にたどる部分集合列挙アルゴリズムにおいて
集合 X の次の出力 X' は次で表される

$$X' = \begin{cases} X \triangle \{1\} & (|X| \text{ 偶数}) \\ X \triangle \{1 + \min X\} & (|X| \text{ 奇数}) \end{cases}$$

証明：演習問題

部分集合列挙アルゴリズム (Gray 符号)

入力： n
出力： $\{1, \ldots, n\}$ **の部分集合すべて**

- $X := \emptyset$, $p := 0$, $i := 0$ **と初期化**
- **以下を繰り返し** (* **不変条件：** $p = |X| \mod 2$**，** $i = \min X$ *)
  - X **を出力**
  - $i = n$ **ならば，アルゴリズムを停止**
  - $p = 0$ **ならば，** $X := X \triangle \{1\}$, $p := 1$, $i := \min X$
  - $p = 1$ **ならば，** $X := X \triangle \{1+i\}$, $p := 0$, $i := \min X$

部分集合列挙アルゴリズム (Gray 符号)

入力： n
出力： $\{1, \ldots, n\}$ **の部分集合すべて**

- $X := \emptyset$, $p := 0$, $i := 0$ **と初期化**
- **以下を繰り返し** (* **不変条件：** $p = |X| \mod 2$**，** $i = \min X$ *)
  - X **を出力**
  - $i = n$ **ならば，アルゴリズムを停止**
  - $p = 0$ **ならば，** $X := X \triangle \{1\}$, $p := 1$, $i := \min X$
  - $p = 1$ **ならば，** $X := X \triangle \{1+i\}$, $p := 0$, $i := \min X$

前述の命題 2 と繰り返しにおいて不変条件が保たれていることから分かる

部分集合列挙アルゴリズム (Gray 符号)

入力： n
出力： $\{1, \ldots, n\}$ **の部分集合すべて**

- $X := \emptyset$, $p := 0$, $i := 0$ **と初期化**
- **以下を繰り返し** (* **不変条件：** $p = |X| \mod 2$**，** $i = \min X$ *)
  - X **を出力**
  - $i = n$ **ならば，アルゴリズムを停止**
  - $p = 0$ **ならば，** $X := X \triangle \{1\}$, $p := 1$, $i := \min X$
  - $p = 1$ **ならば，** $X := X \triangle \{1+i\}$, $p := 0$, $i := \min X$

- **時間**
  - **対称差を取る操作と最小値を計算する操作にかかる時間は単純なデータ構造で** $\mathrm{O}(1)$
  - **集合** 1 **つの出力に** $\mathrm{O}(n)$ **必要**
  - $\therefore$ **最悪時遅延が** $\mathrm{O}(n)$
- **領域**
  - X, p, i **のサイズの和で，これは** $\mathrm{O}(n)$

コンパクト出力 (compact output)

「コンパクト出力」の問題意識

- 要素数が高々nの集合を出力しようと思うと，
  出力だけでどうしても$\Theta(n)$だけ計算量が必要となる
- うまく出力を圧縮できないか?

コンパクト出力 (compact output)

「コンパクト出力」の問題意識

- 要素数が高々nの集合を出力しようと思うと，
  出力だけでどうしても$\Theta(n)$だけ計算量が必要となる
- うまく出力を圧縮できないか?

「コンパクト出力」の行なうこと

- 出力の圧縮
- すべてを出力してから圧縮するのではなくて，
  圧縮した出力をする

コンパクト出力の例

- 差分出力 ← この講義ではこれのみ扱う
- 履歴出力
- 2 分決定図 (BDD) 出力

部分集合列挙問題 — 差分出力の例

$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{2,3\}$, $\{1,2,3\}$,

$\{1,3\}$, $\{3\}$, $\{3,4\}$, $\{1,3,4\}$, $\{1,2,3,4\}$, $\{2,3,4\}$,

$\{2,4\}$, $\{1,2,4\}$, $\{1,4\}$, $\{4\}$

部分集合列挙問題 — 差分出力の例

$\emptyset$,
$\{1\}$, $+1$,
$\{1,2\}$, $+2$,
$\{2\}$, -1,
$\{2,3\}$, $+3$,
$\{1,2,3\}$, $+1$,
$\{1,3\}$, -2,
$\{3\}$, -1,
$\{3,4\}$, $+4$,
$\{1,3,4\}$, $+1$,
$\{1,2,3,4\}$, $+2$,
$\{2,3,4\}$, -1,
$\{2,4\}$, -3,
$\{1,2,4\}$, -1,
$\{1,4\}$, -2,
$\{4\}$ -1

部分集合列挙アルゴリズム (Gray 符号，差分出力)

入力： n
出力： $\{1, \ldots, n\}$ **の部分集合すべて**

- $X := \emptyset$, $p := 0$, $i := 0$ **と初期化して，** X **を出力**
- **以下を繰り返し** (* **不変条件：** $p = |X| \mod 2$**，** $i = \min X$ *)
  - $i = n$ **ならば，アルゴリズムを停止**
  - $p = 0$ **ならば，**
    - $1 \in X$ **ならば，「**-1**」を出力**
    - $1 \notin X$ **ならば，「**$+1$**」を出力**
    - $X := X \triangle \{1\}$, $p := 1$, $i := \min X$
  - $p = 1$ **ならば，**
    - $1+i \in X$ **ならば，「**$-(1+i)$**」を出力**
    - $1+i \notin X$ **ならば，「**$+(1+i)$**」を出力**
    - $X := X \triangle \{1+i\}$, $p := 0$, $i := \min X$

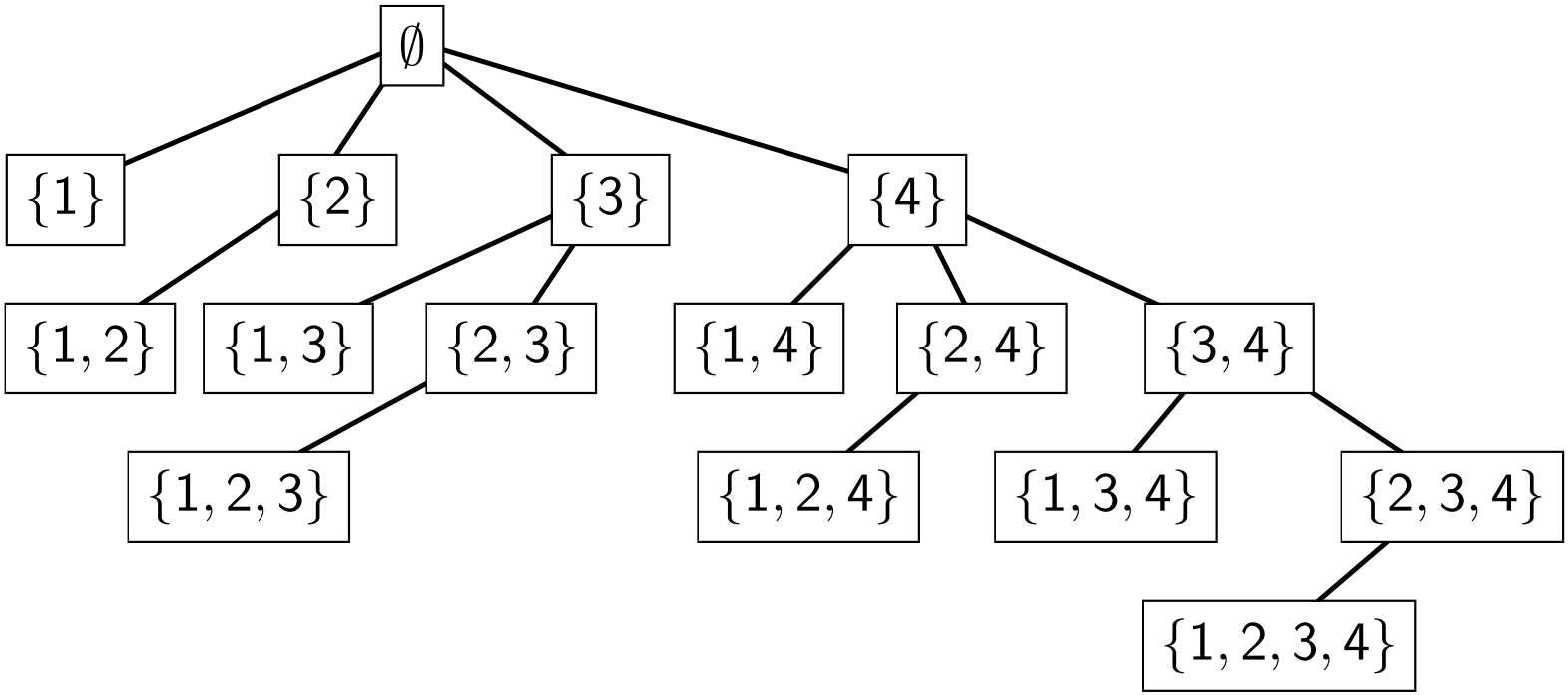
時間計算量：最悪時遅延 $O(1)$**，領域計算量：** $O(n)$

パート 1 の内容

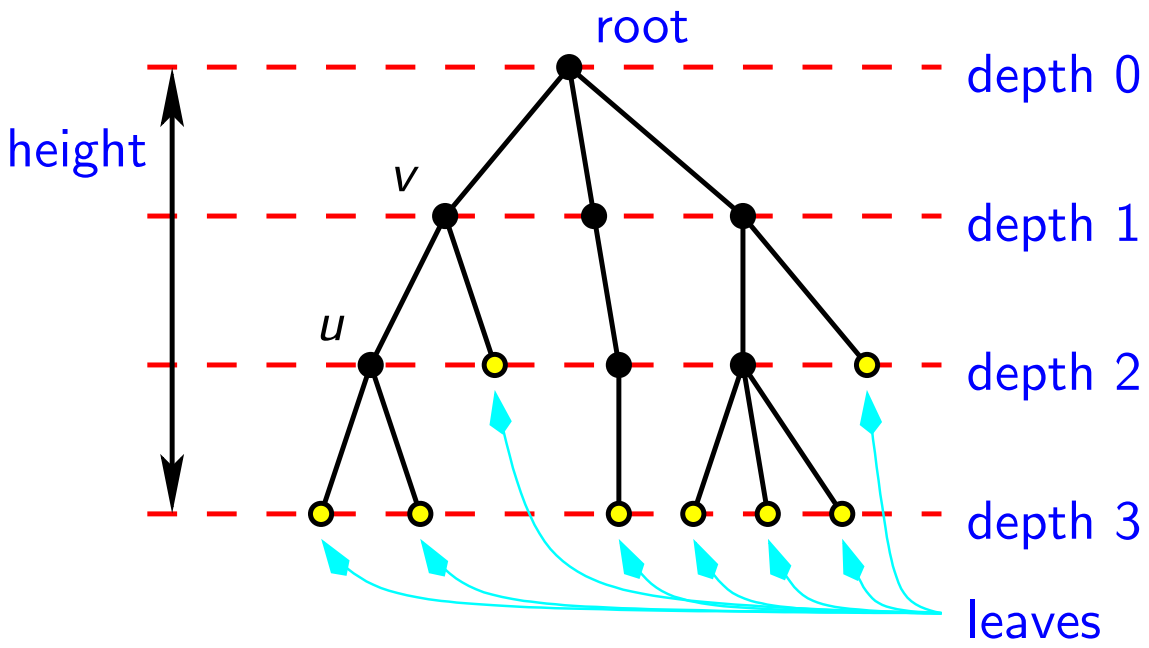
- **列挙問題，列挙アルゴリズムとは?**
- **列挙問題，列挙アルゴリズム設計の難しさ**
- **列挙アルゴリズム設計技法**
  - **分割探索** (binary partition)
  - **組合せ** Gray **符号** (combinatorial Gray code)
  - **逆探索** (reverse search)
- **難しい列挙問題**

基本的な考え方

- 部分集合全体の集合上に根付き木を定義
- その根付き木を深さ優先探索することで列挙



復習：根付き木の用語



- u は v の子，v は u の親 (その他，親族関係と類似した用語)
- 根付き木の根 (root)：親のいない唯一の頂点
- 根付き木の葉 (leaf)：子のいない頂点
- 頂点 v の深さ (depth)：根から v への唯一の道の辺数
- 根付き木の高さ (height)：深さの最大値

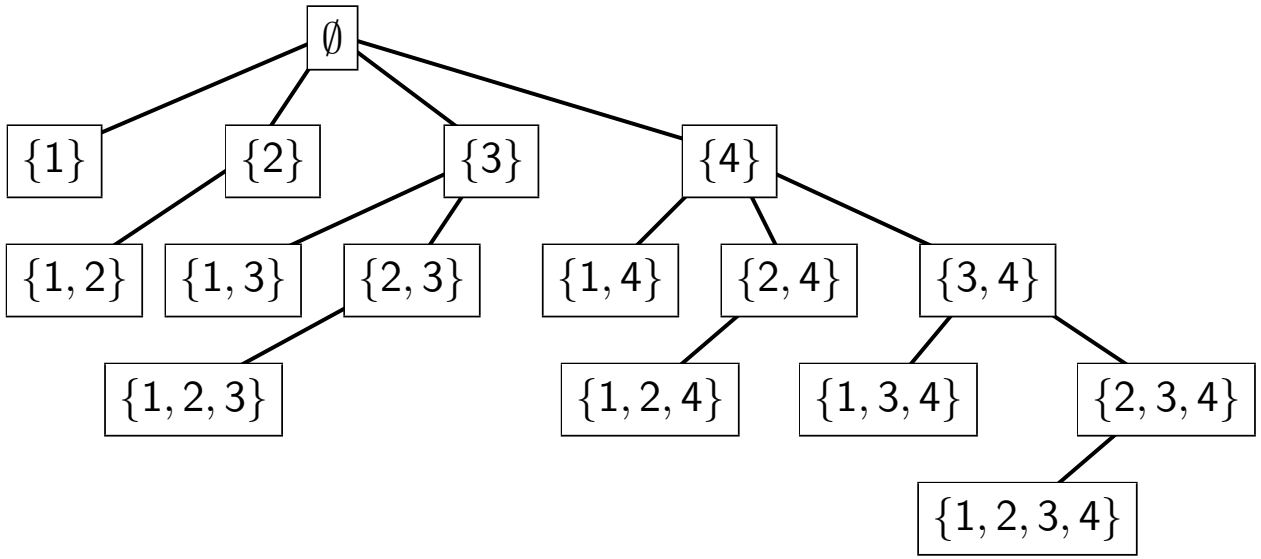
根付き木の構成

- $\emptyset$ を根とする
- 部分集合 $X \subseteq \{1, \ldots, n\}$ $(X \neq \emptyset)$ の親 $p(X)$ を $p(X) := X \setminus \{\min X\}$ と定義

命題 3

この根付き木は well-defined

略証：子と親の要素数を見ると親が子より 1 小さいので



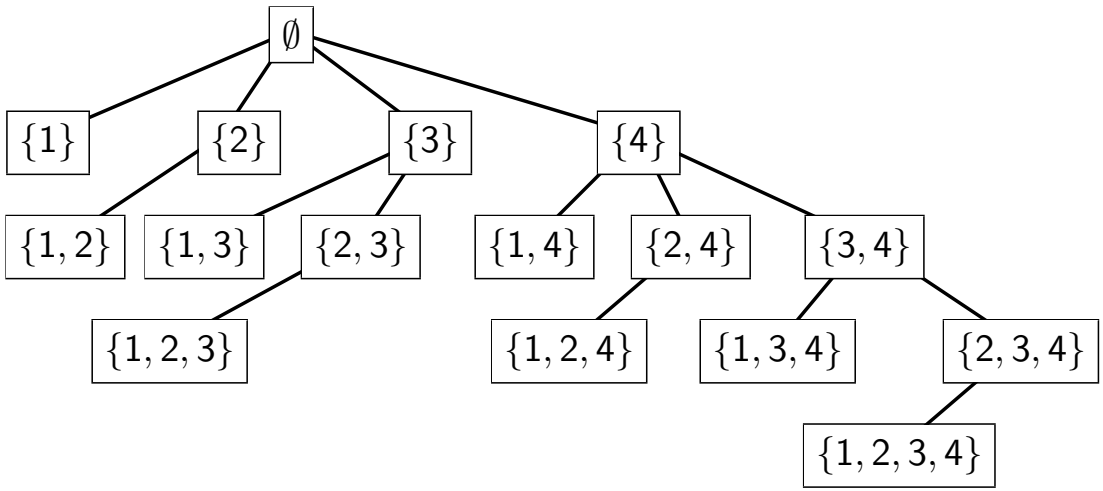
子から親を得る操作

部分集合 $X \subseteq \{1, \ldots, n\}$ $(X \neq \emptyset)$ **の親** $p(X)$ **を**
$p(X) := X \setminus \{\min X\}$ **と定義**

深さ優先探索では「親から子を得る操作」も必要

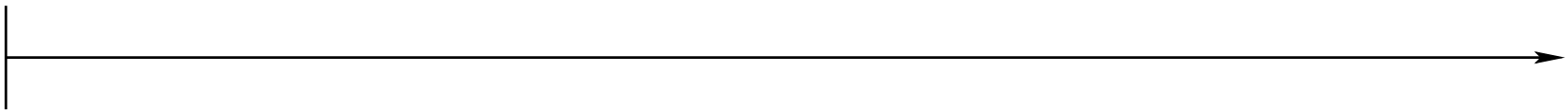
親から子を得る操作

部分集合 $Y \subseteq \{1, \ldots, n\}$ **と任意の** $i < \min Y$ **に対して**
$Y \cup \{i\}$ **は** Y **の子で (ただし，**$\min \emptyset = \infty$**) ，**
Y **の子はこのように表現されるものに限る**



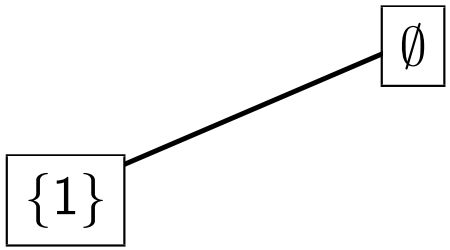
アルゴリズムの動作を見てみる

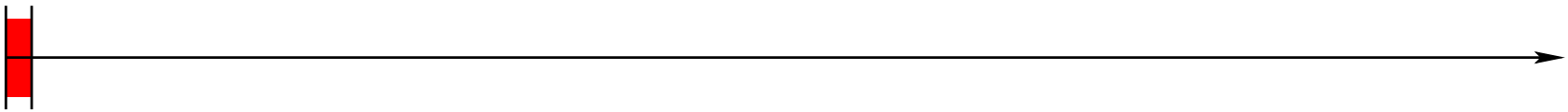
$\boxed{\emptyset}$



$\emptyset$

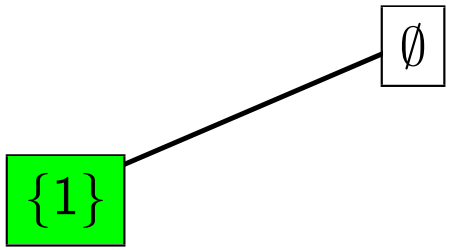
アルゴリズムの動作を見てみる

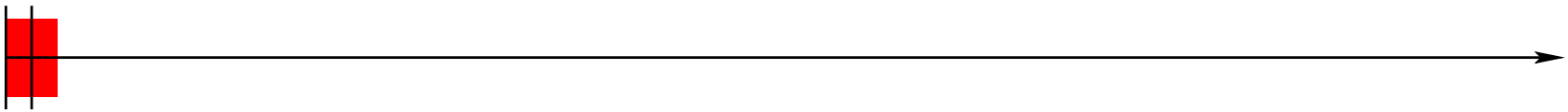




∅, {1}

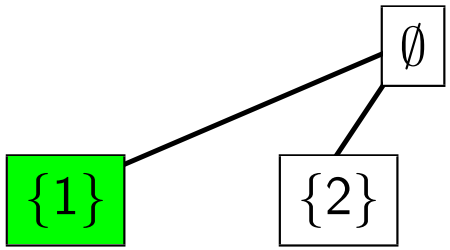
アルゴリズムの動作を見てみる

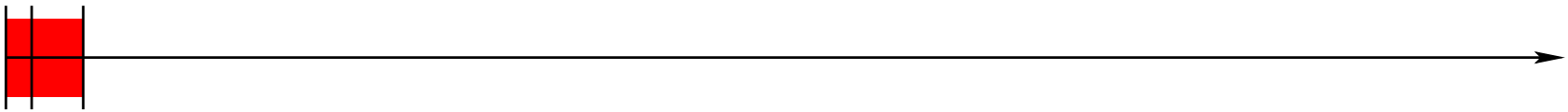




∅, {1}

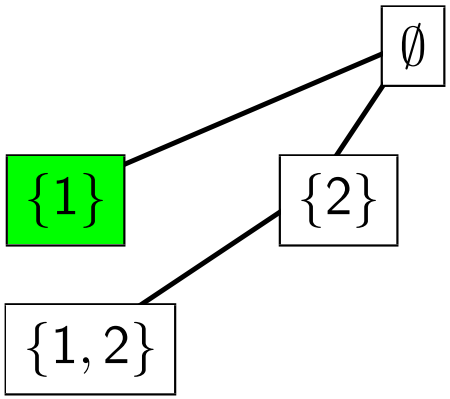
アルゴリズムの動作を見てみる

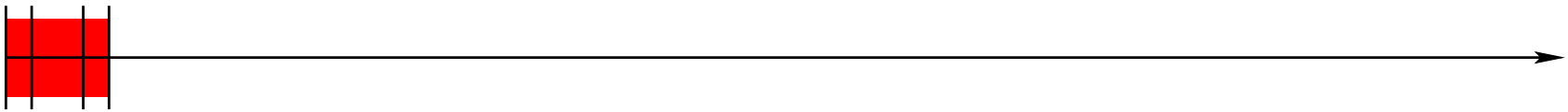




$\emptyset, \{1\}, \{2\}$

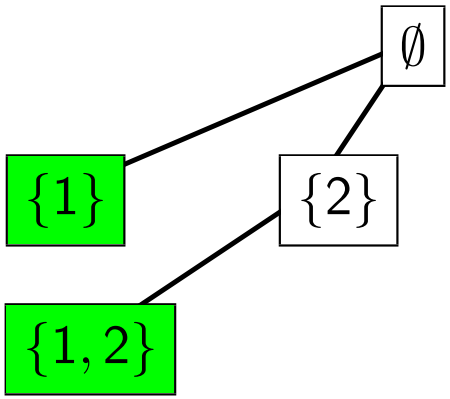
アルゴリズムの動作を見てみる

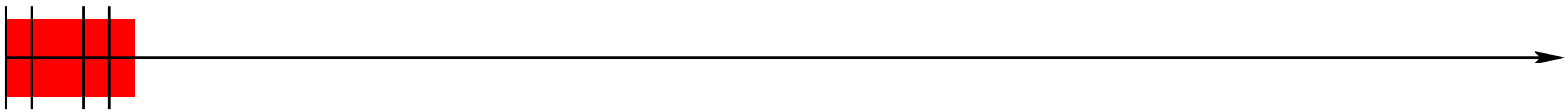




$\emptyset, \{1\}, \{2\}, \{1,2\}$

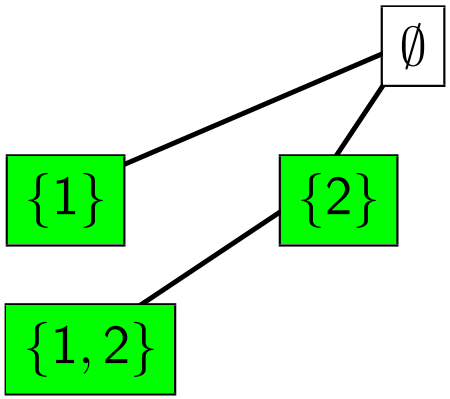
アルゴリズムの動作を見てみる

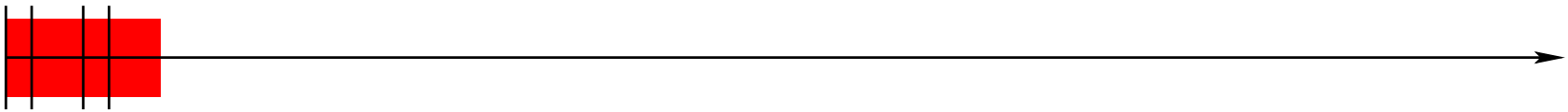




$\emptyset, \{1\}, \{2\}, \{1, 2\}$

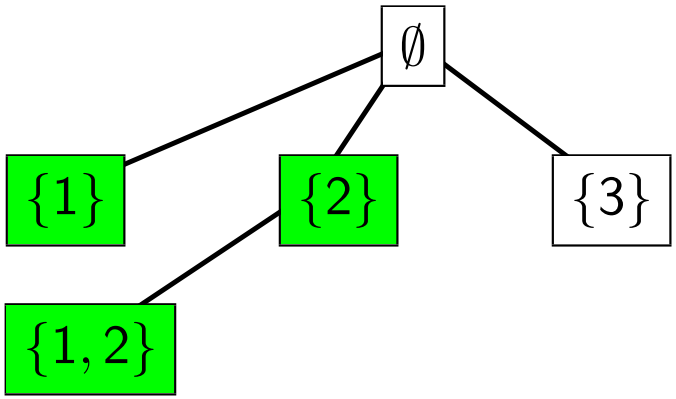
アルゴリズムの動作を見てみる

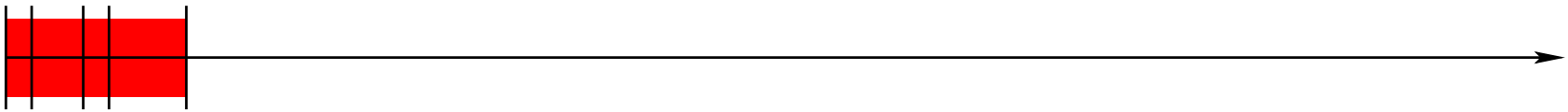




$\emptyset, \{1\}, \{2\}, \{1, 2\}$

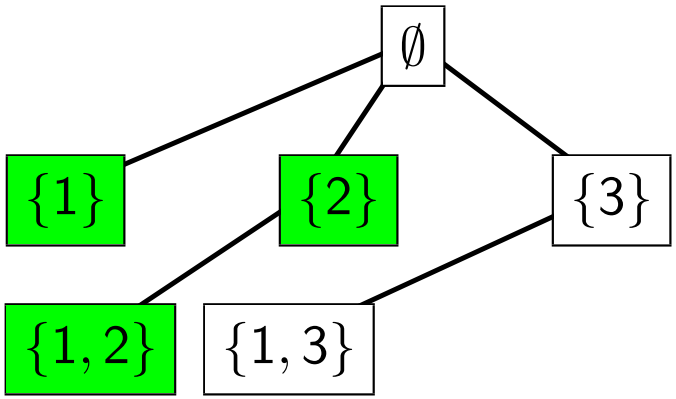
アルゴリズムの動作を見てみる

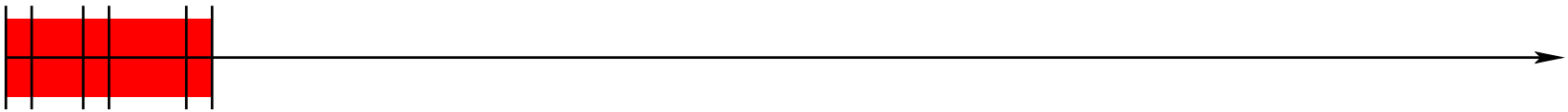




$\emptyset, \{1\}, \{2\}, \{1,2\}, \{3\}$

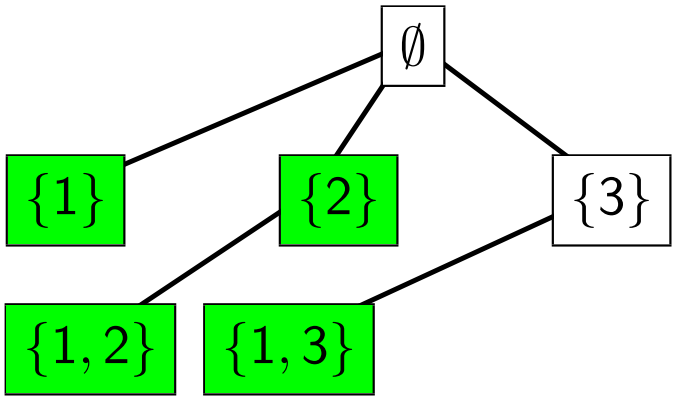
アルゴリズムの動作を見てみる

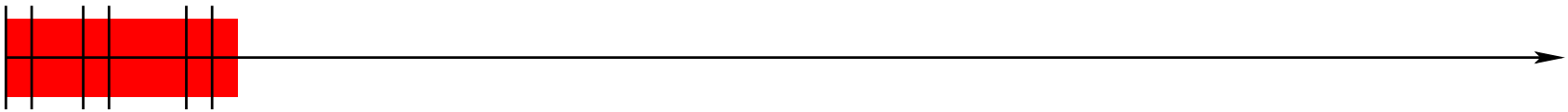




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$

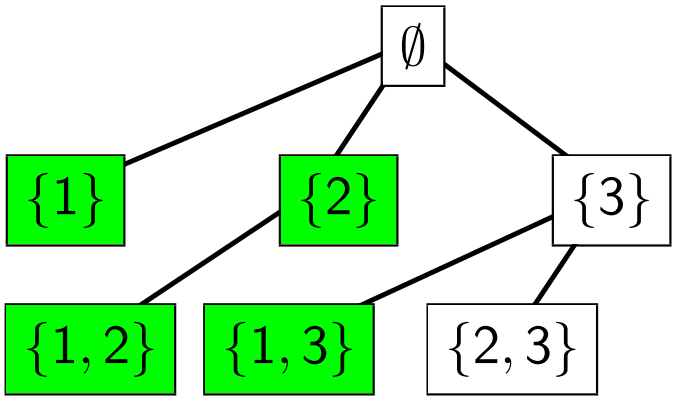
アルゴリズムの動作を見てみる

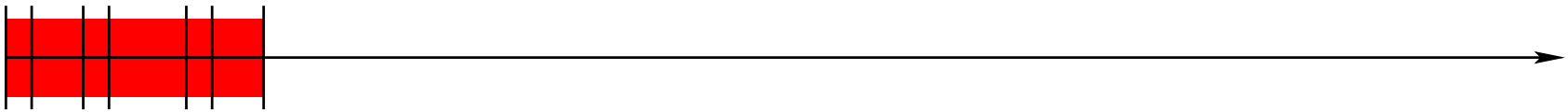




$\emptyset, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}$

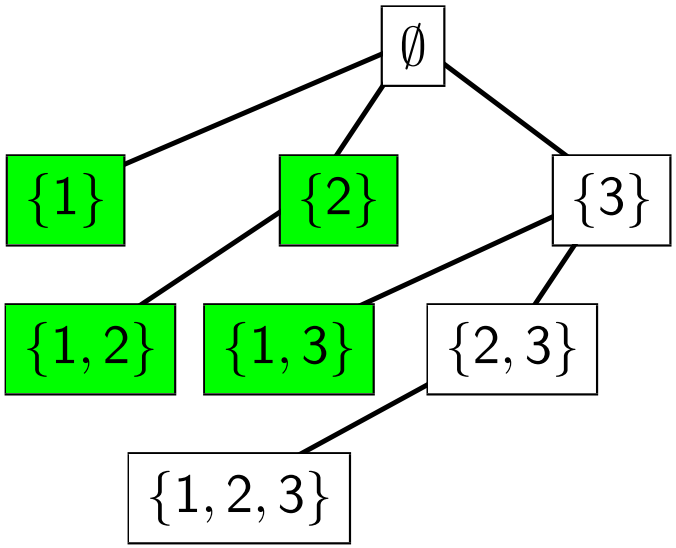
アルゴリズムの動作を見てみる

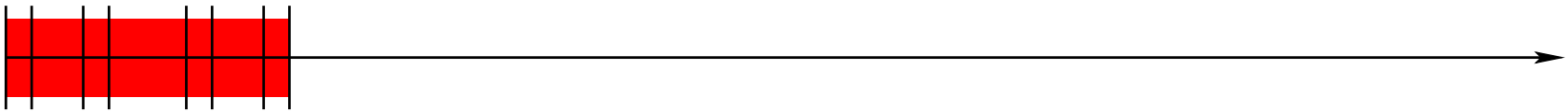




∅, {1}, {2}, {1, 2}, {3}, {1, 3}, {2, 3}

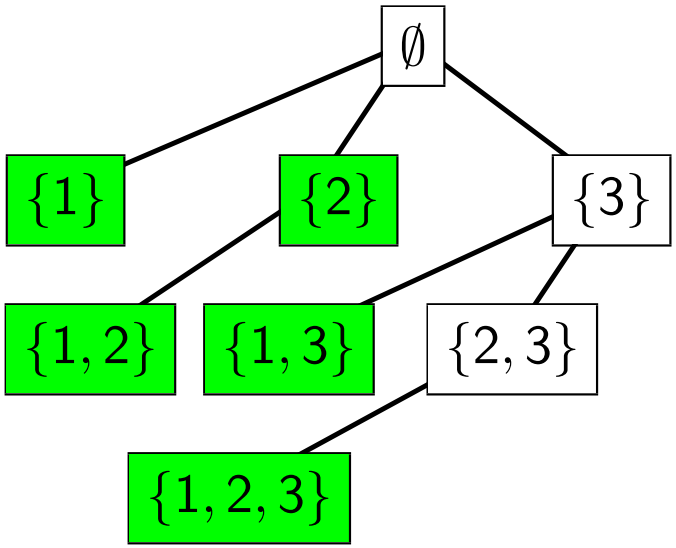
アルゴリズムの動作を見てみる

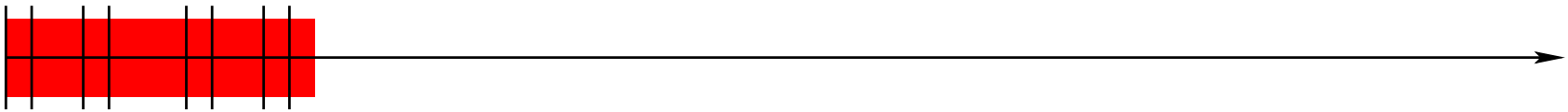




$\emptyset,\ \{1\},\ \{2\},\ \{1,2\},\ \{3\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\}$

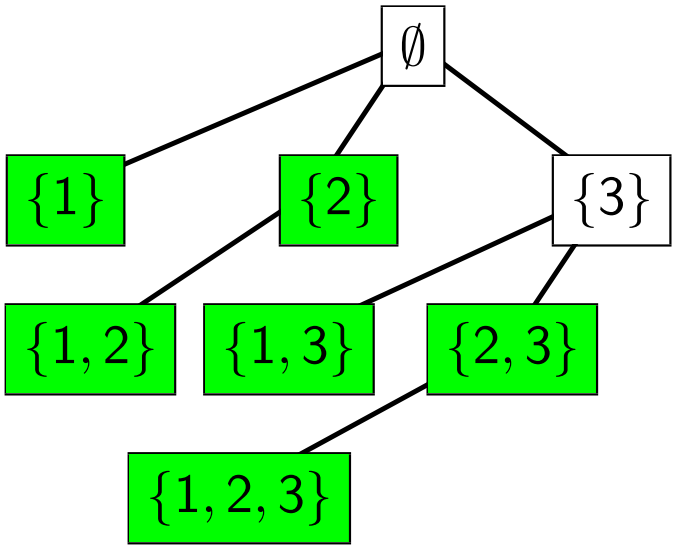
アルゴリズムの動作を見てみる

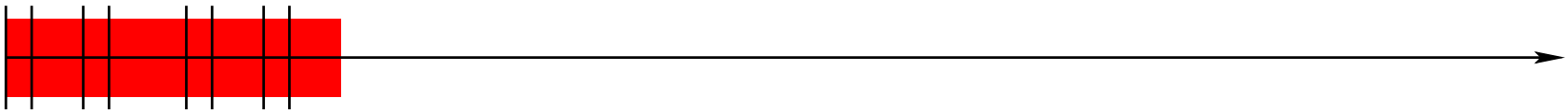




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$

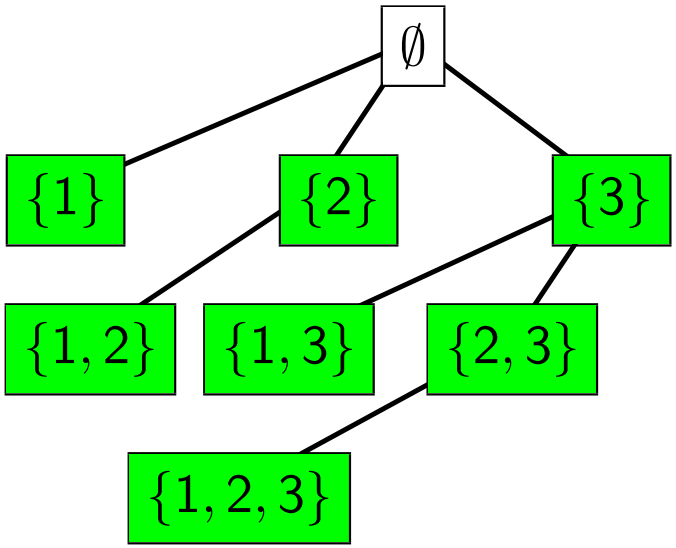
アルゴリズムの動作を見てみる

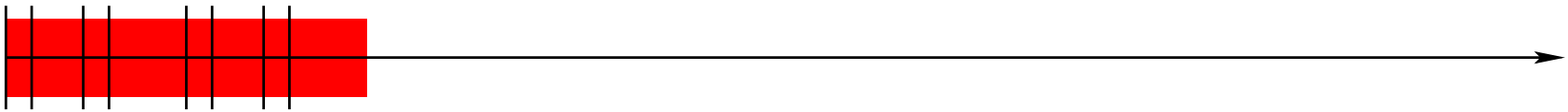




$\emptyset,\ \{1\},\ \{2\},\ \{1,2\},\ \{3\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\}$

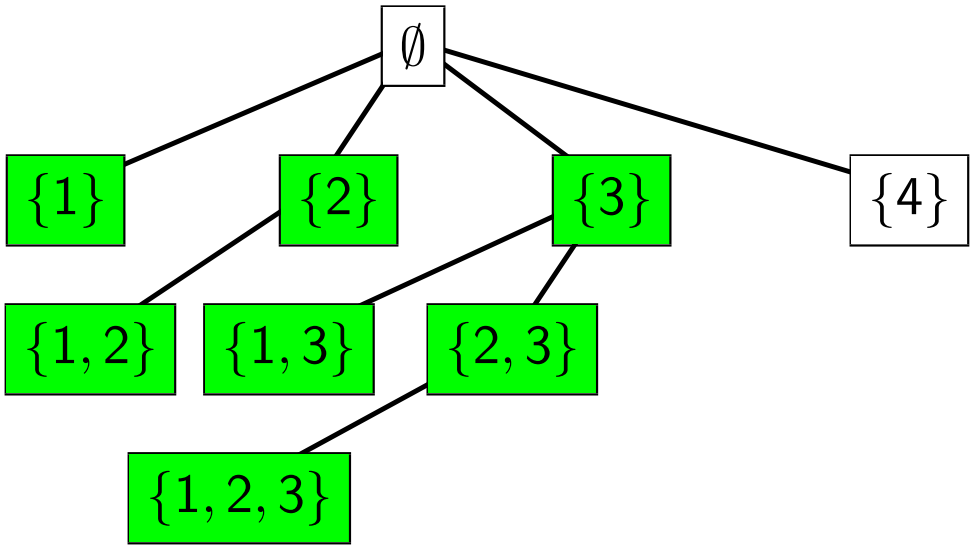
アルゴリズムの動作を見てみる

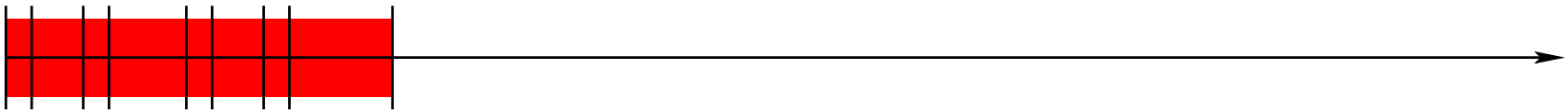




$\emptyset,\ \{1\},\ \{2\},\ \{1,2\},\ \{3\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\}$

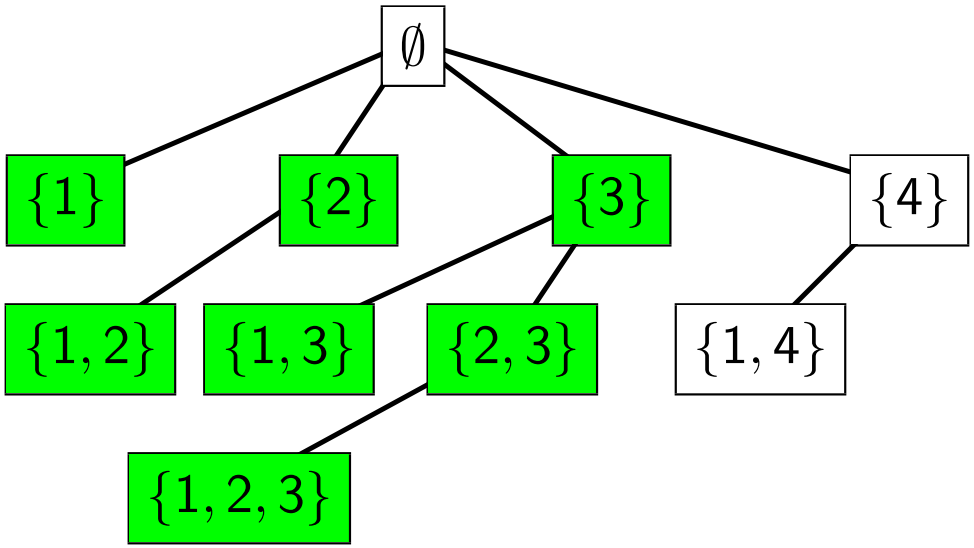
アルゴリズムの動作を見てみる

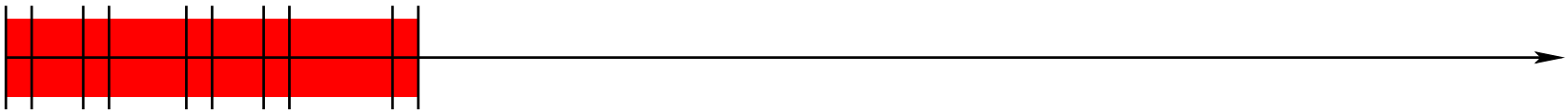




$\emptyset,\ \{1\},\ \{2\},\ \{1,2\},\ \{3\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\},\ \{4\}$

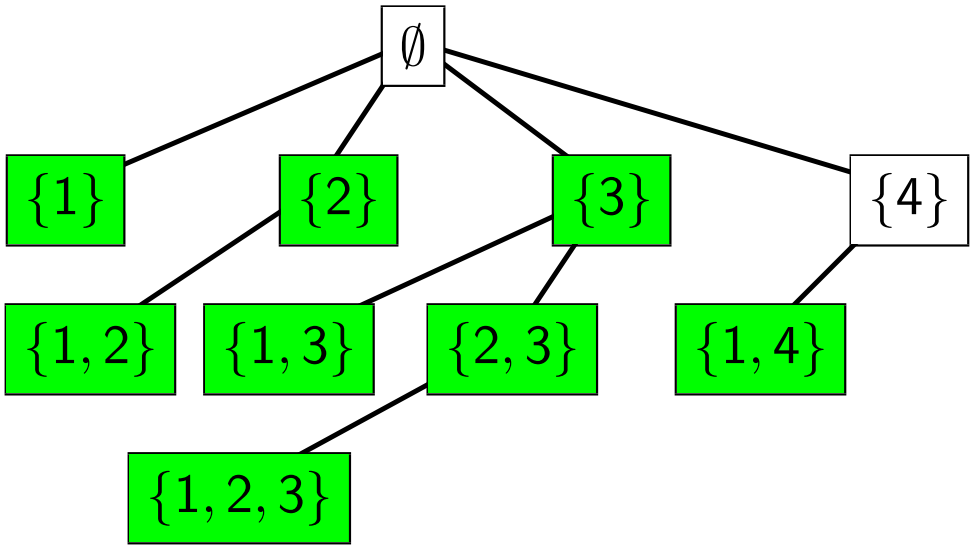
アルゴリズムの動作を見てみる

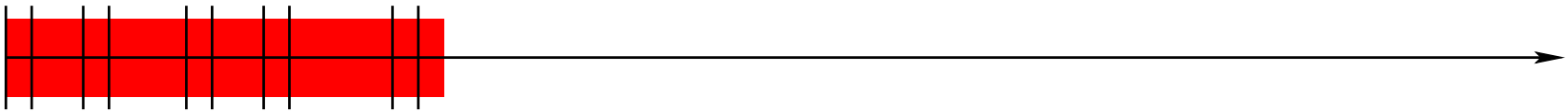




$\emptyset,\ \{1\},\ \{2\},\ \{1,2\},\ \{3\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\},\ \{4\},\ \{1,4\}$

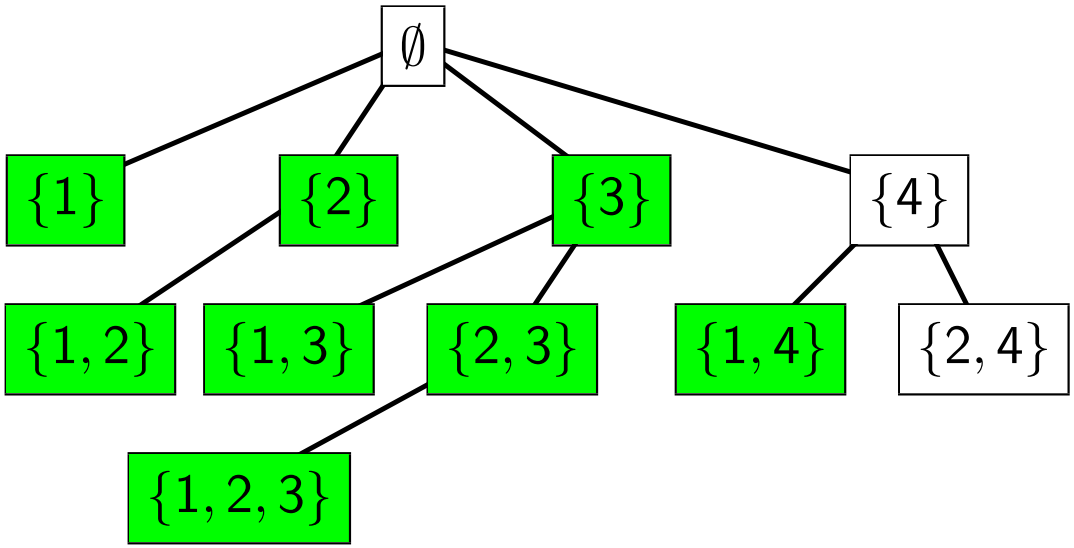
アルゴリズムの動作を見てみる

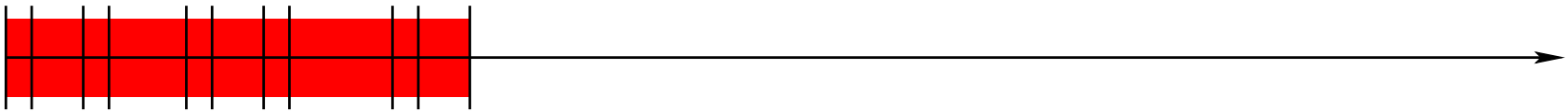




$\emptyset,\ \{1\},\ \{2\},\ \{1,2\},\ \{3\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\},\ \{4\},\ \{1,4\}$

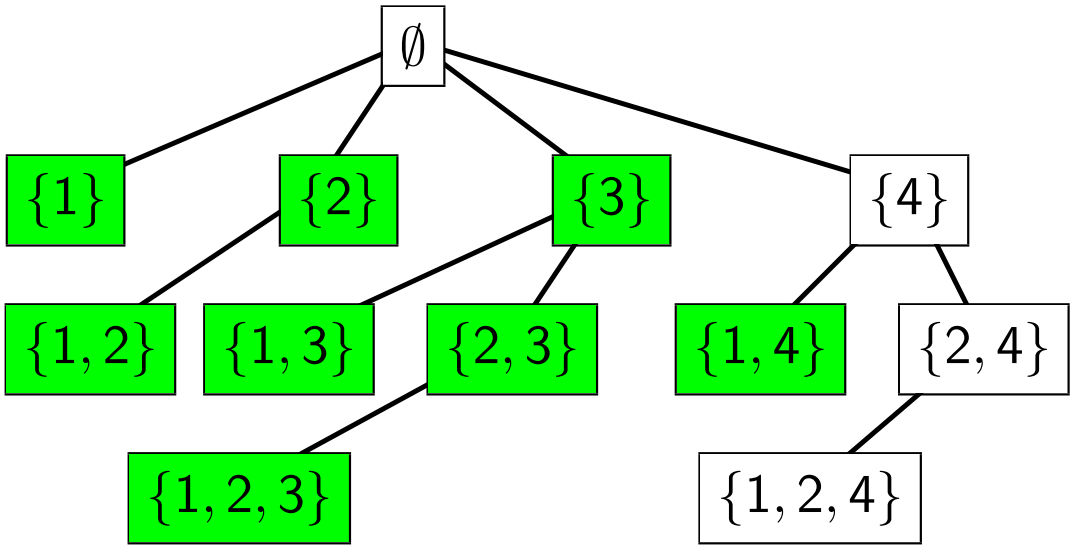
アルゴリズムの動作を見てみる

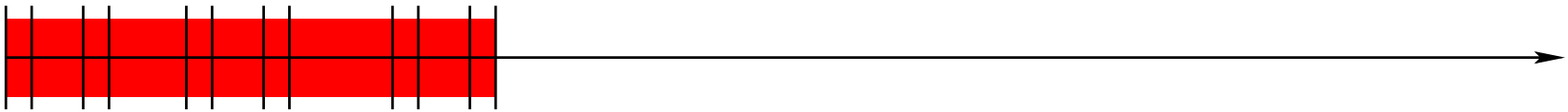




∅, {1}, {2}, {1, 2}, {3}, {1, 3}, {2, 3}, {1, 2, 3}, {4}, {1, 4}, {2, 4}

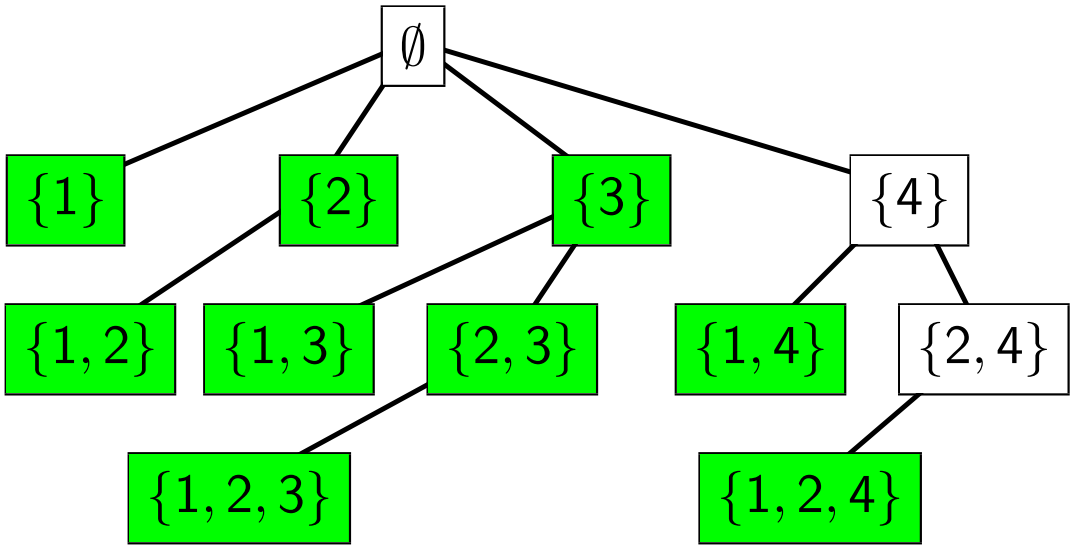
アルゴリズムの動作を見てみる

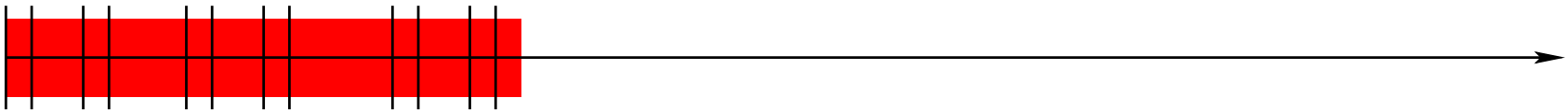




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$

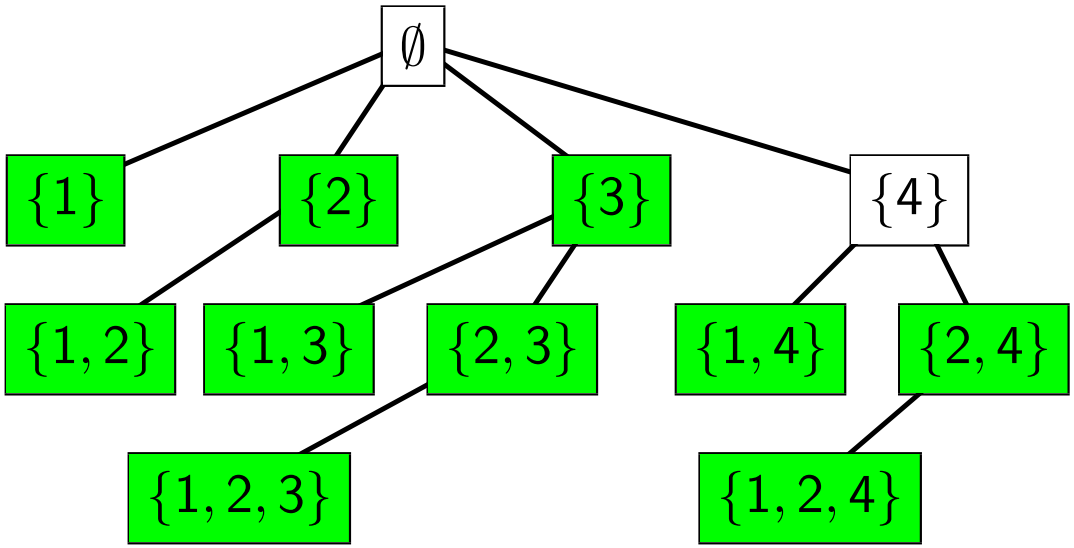
アルゴリズムの動作を見てみる

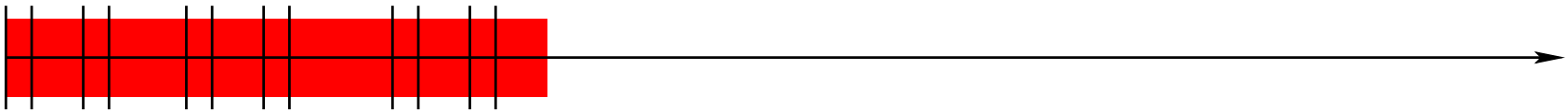




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$

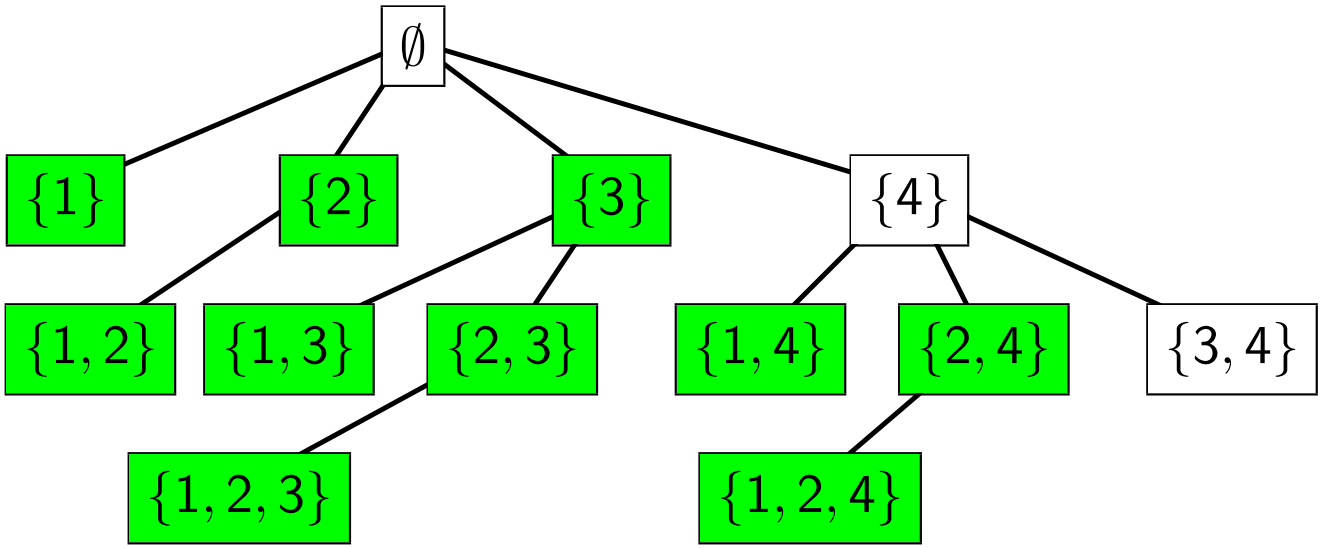
アルゴリズムの動作を見てみる

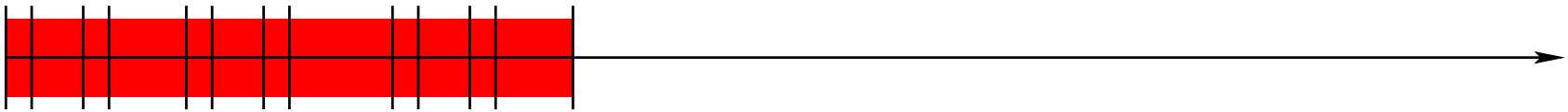




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$

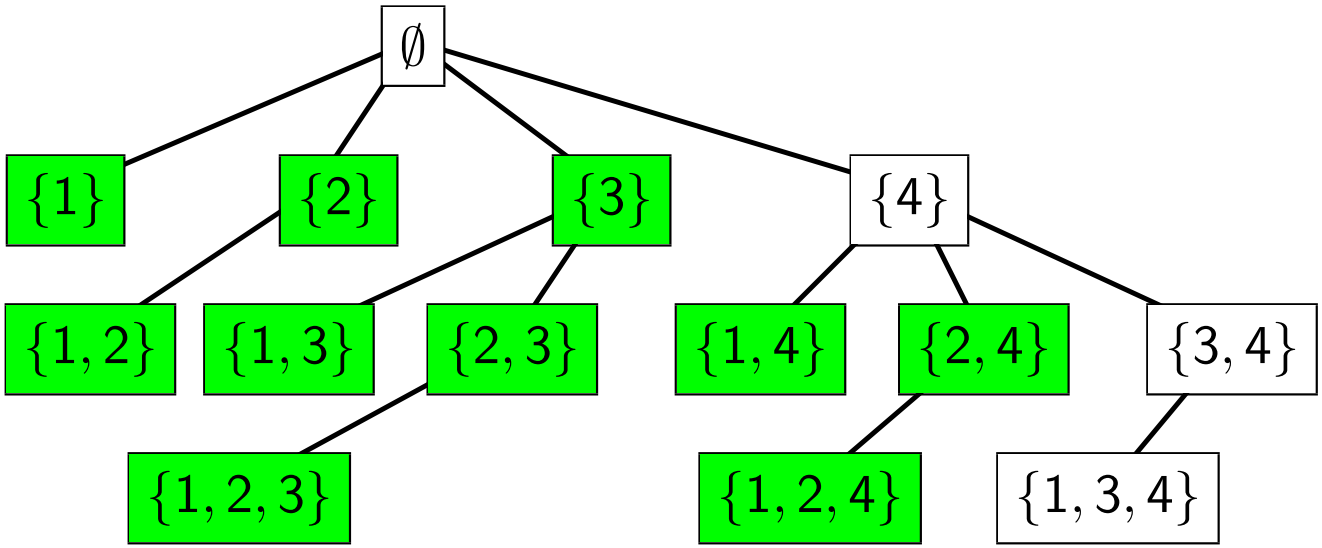
アルゴリズムの動作を見てみる

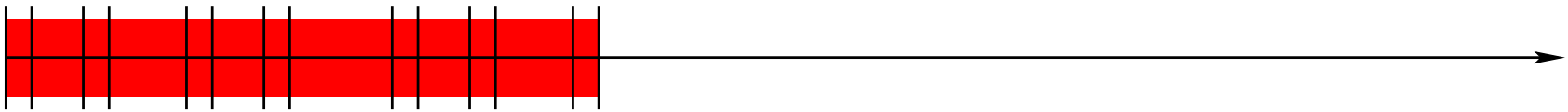




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$

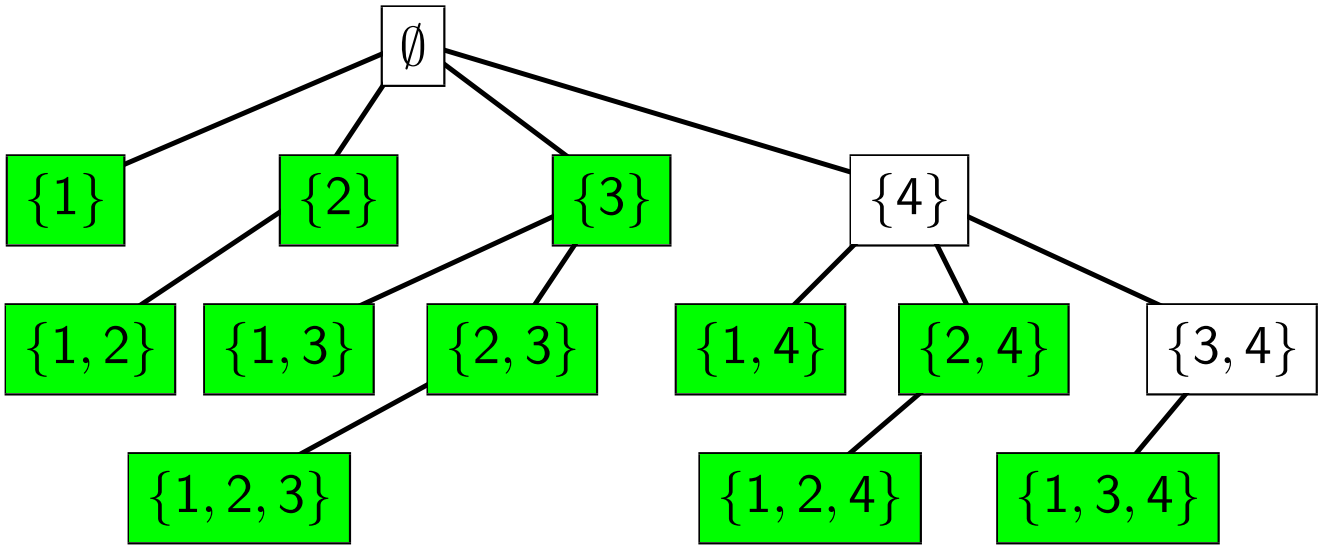
アルゴリズムの動作を見てみる

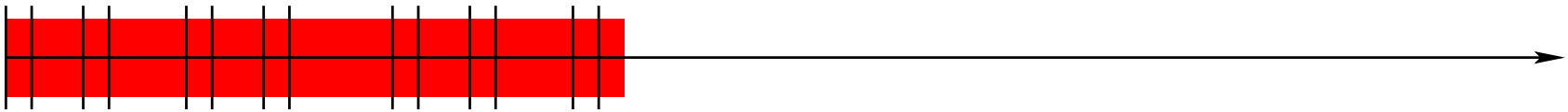




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$

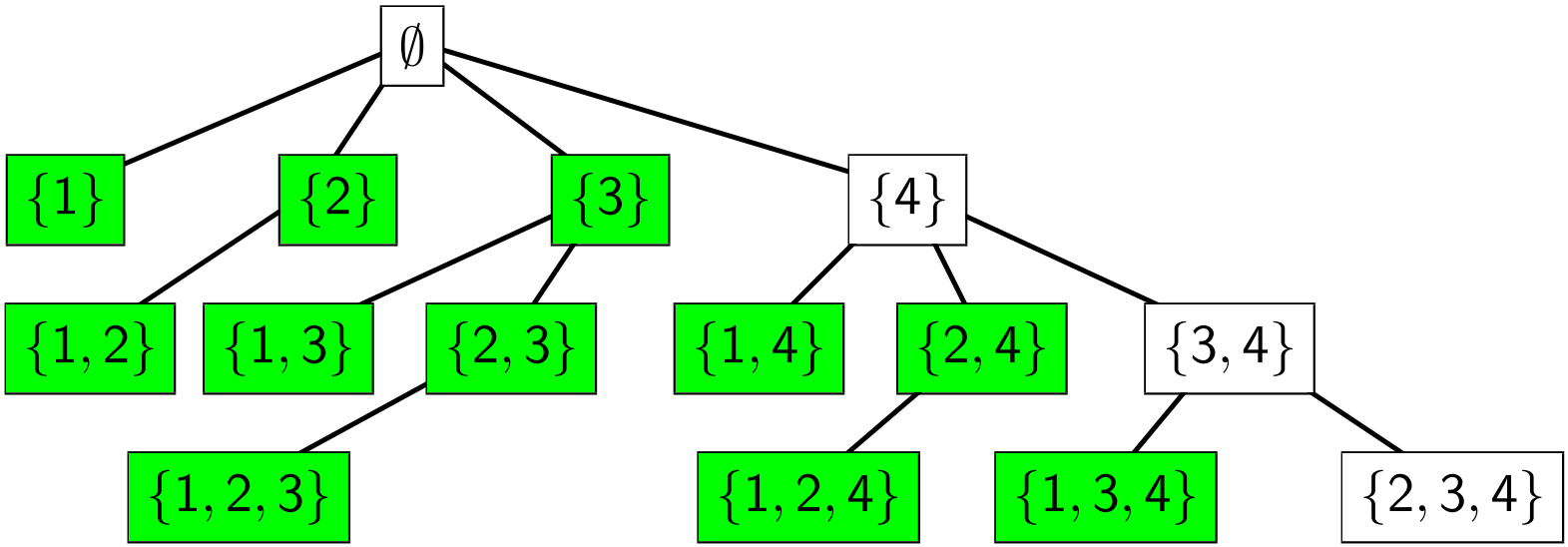
アルゴリズムの動作を見てみる

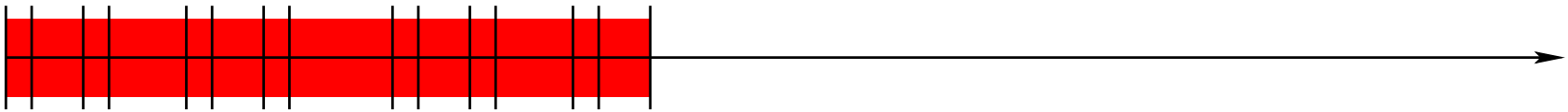




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$

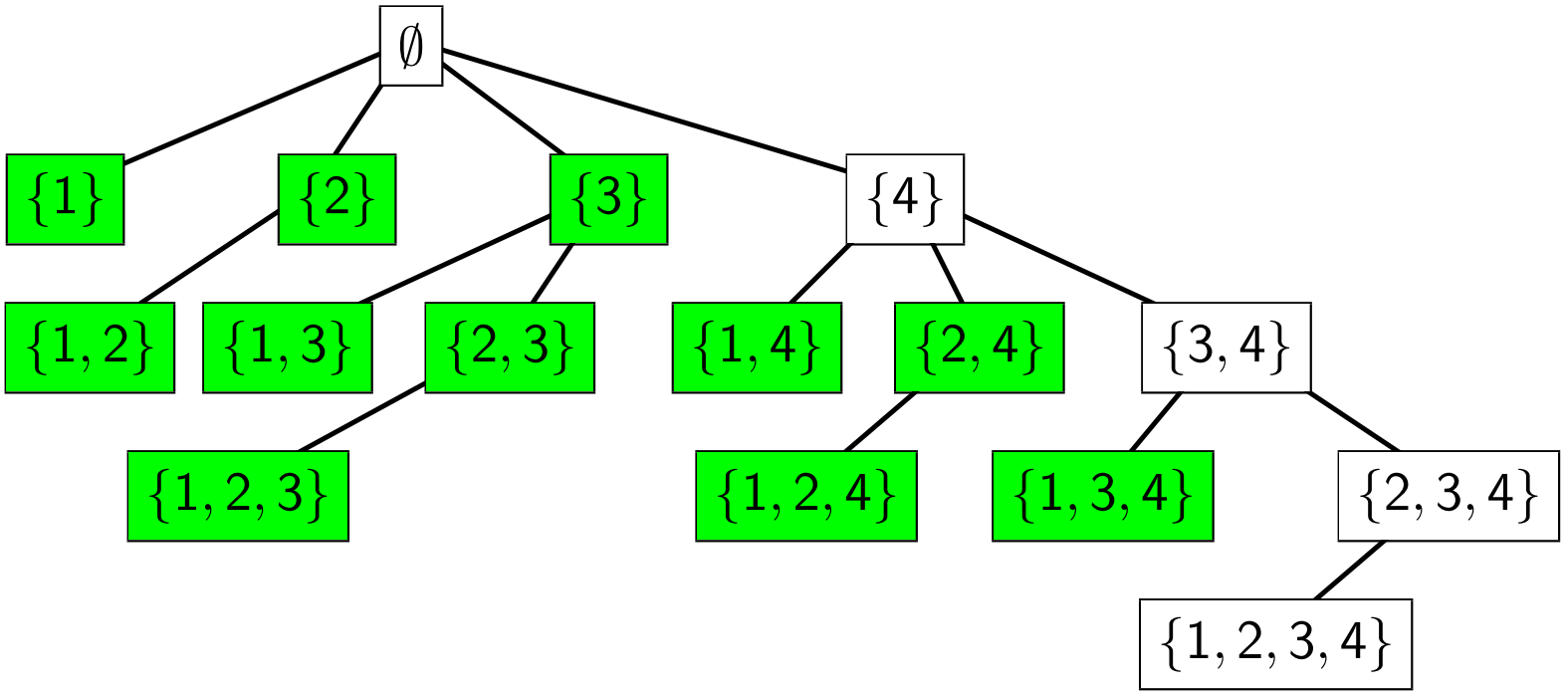
アルゴリズムの動作を見てみる

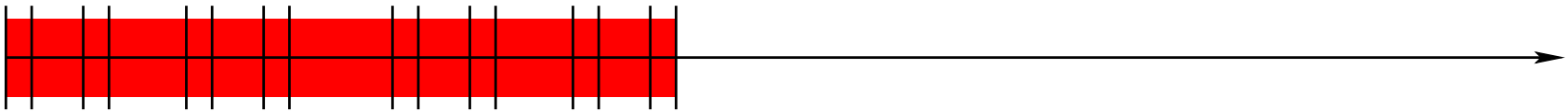




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{2,3,4\}$

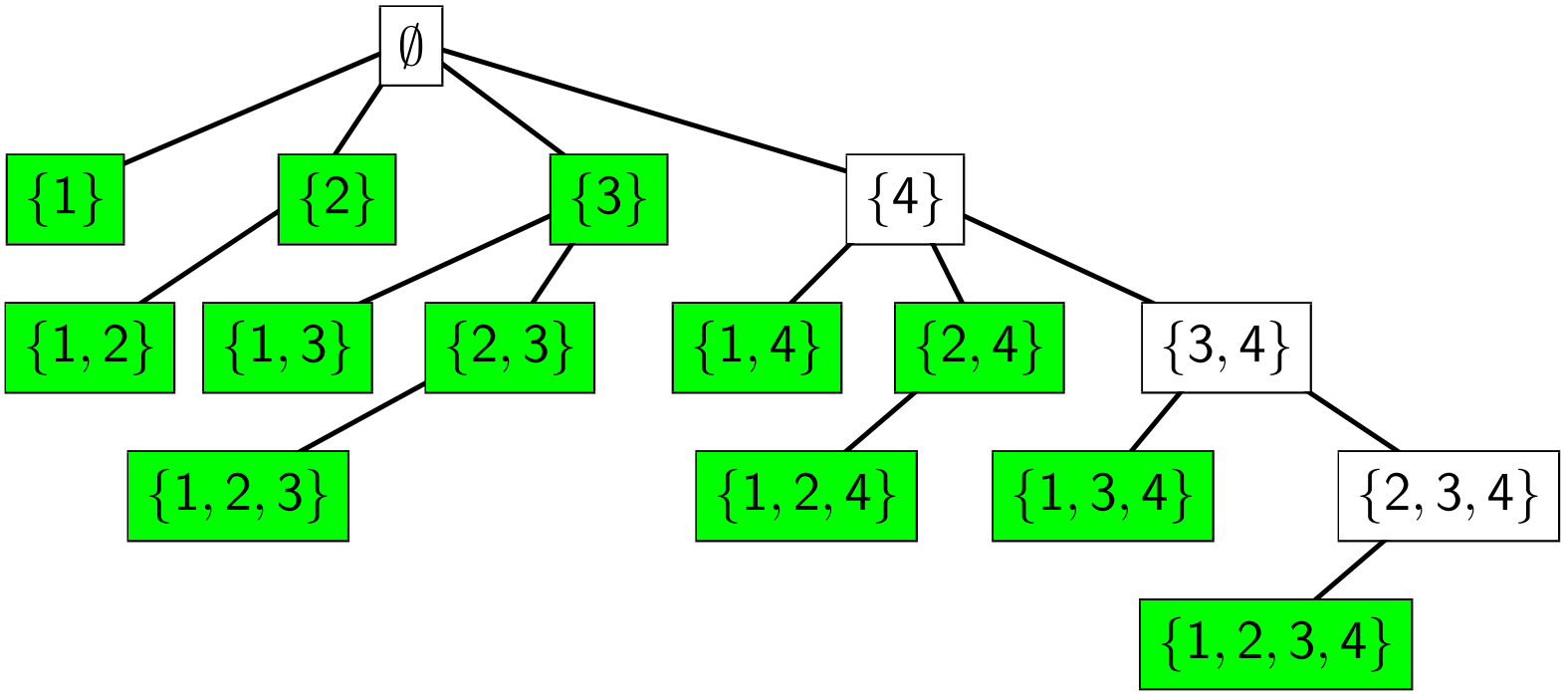
アルゴリズムの動作を見てみる

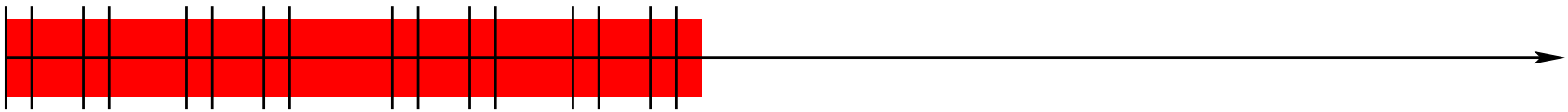




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{2,3,4\}$, $\{1,2,3,4\}$

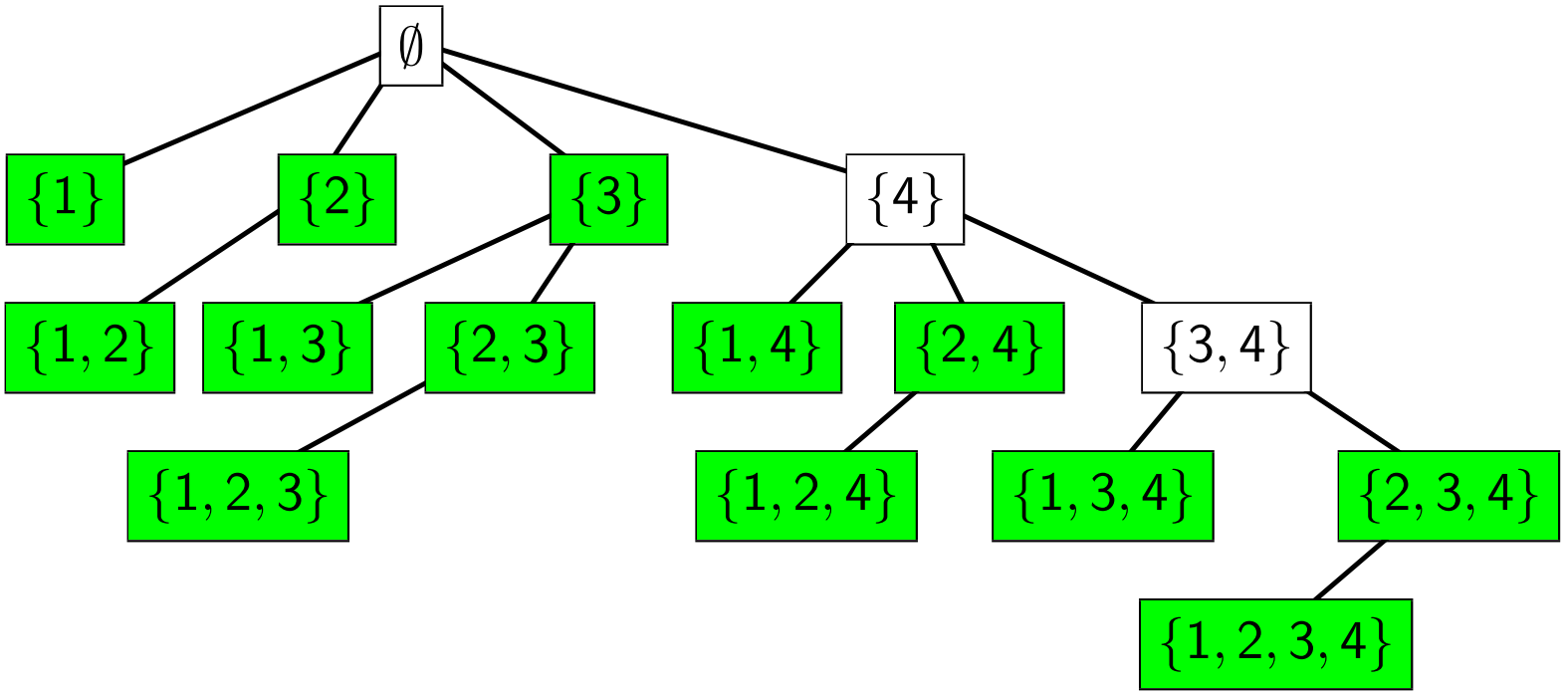
アルゴリズムの動作を見てみる

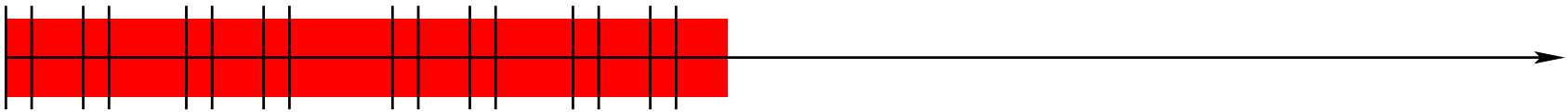




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{2,3,4\}$, $\{1,2,3,4\}$

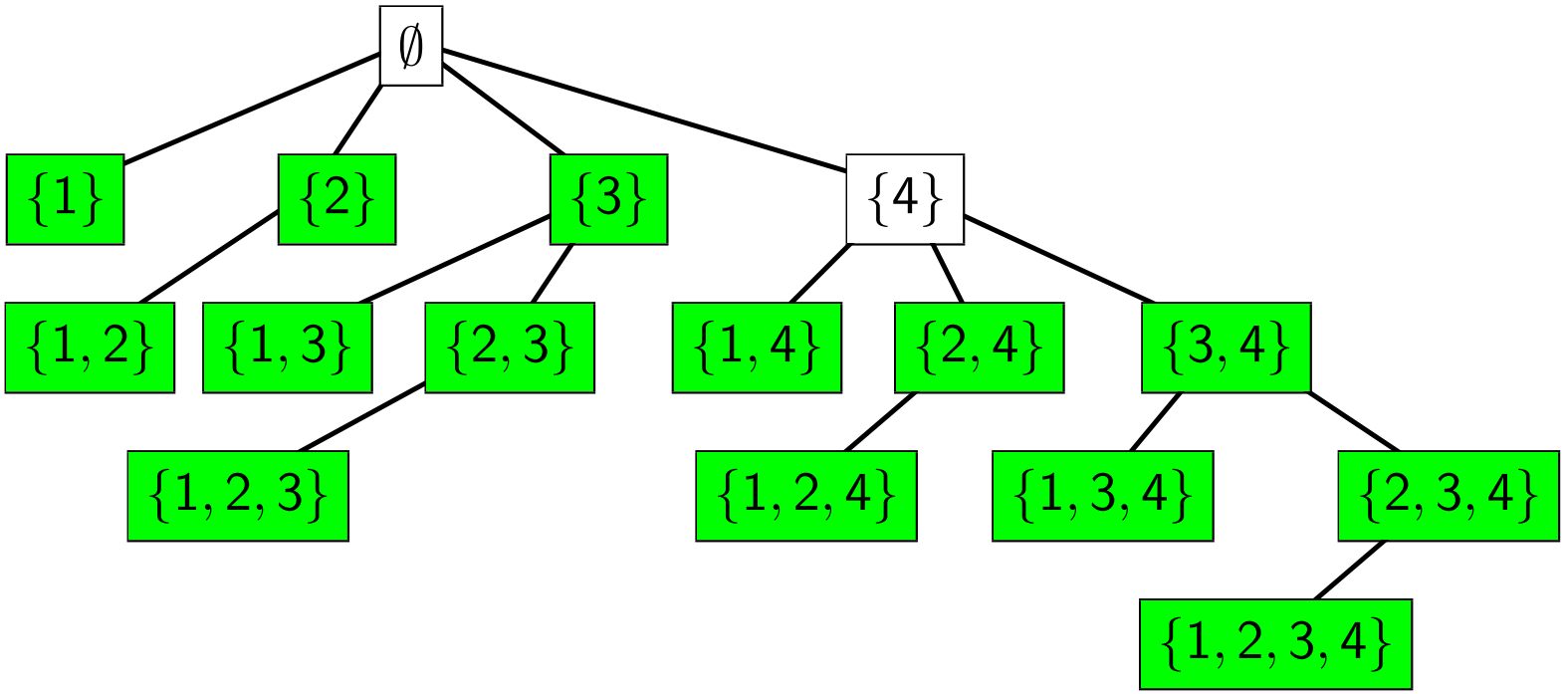
アルゴリズムの動作を見てみる

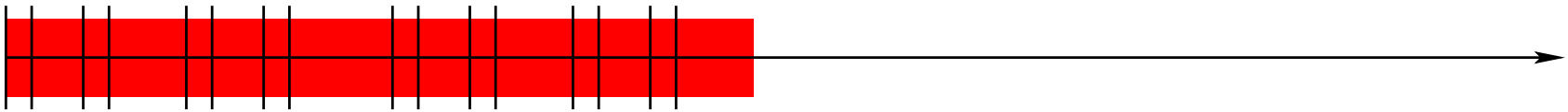




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{2,3,4\}$, $\{1,2,3,4\}$

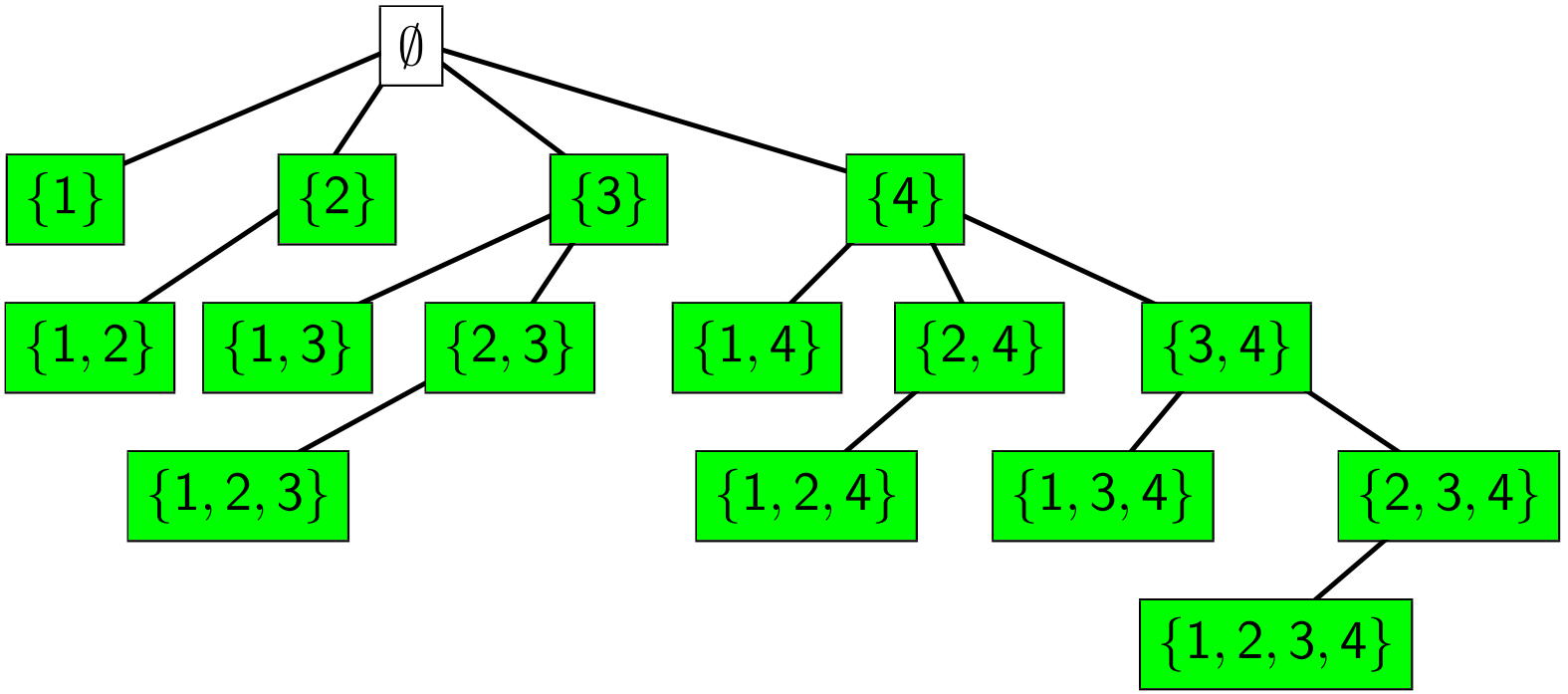
アルゴリズムの動作を見てみる

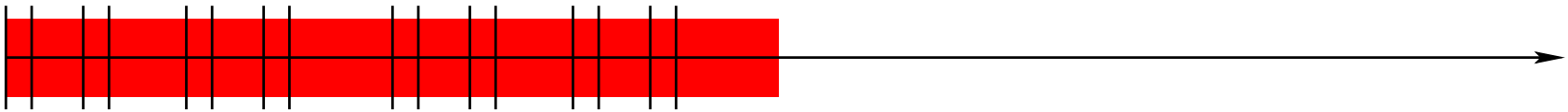




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{2,3,4\}$, $\{1,2,3,4\}$

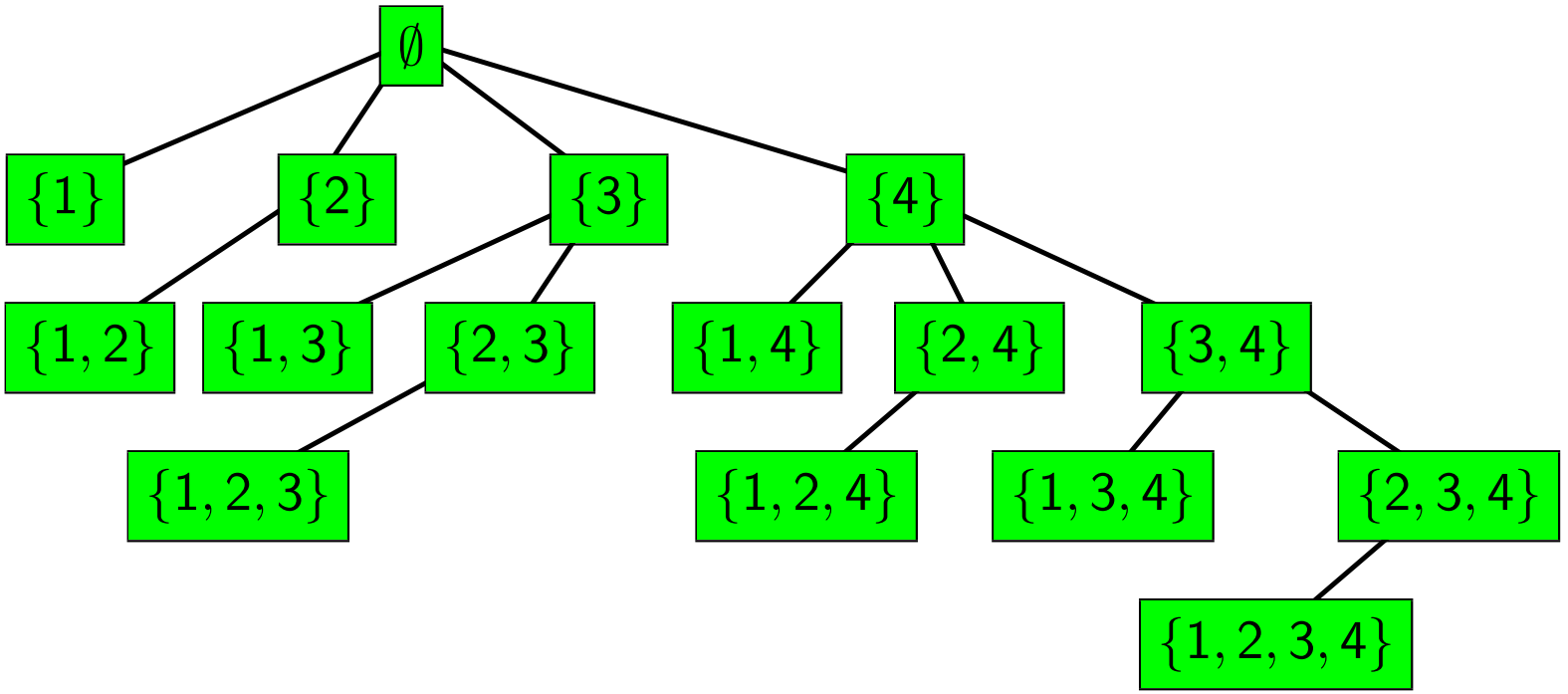
アルゴリズムの動作を見てみる

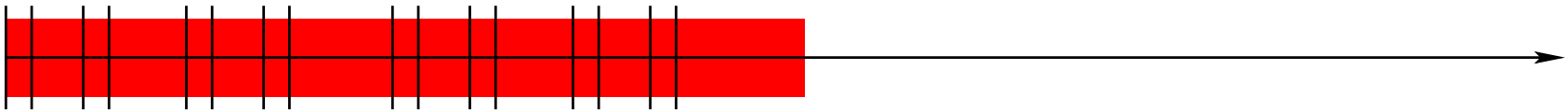




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{2,3,4\}$, $\{1,2,3,4\}$

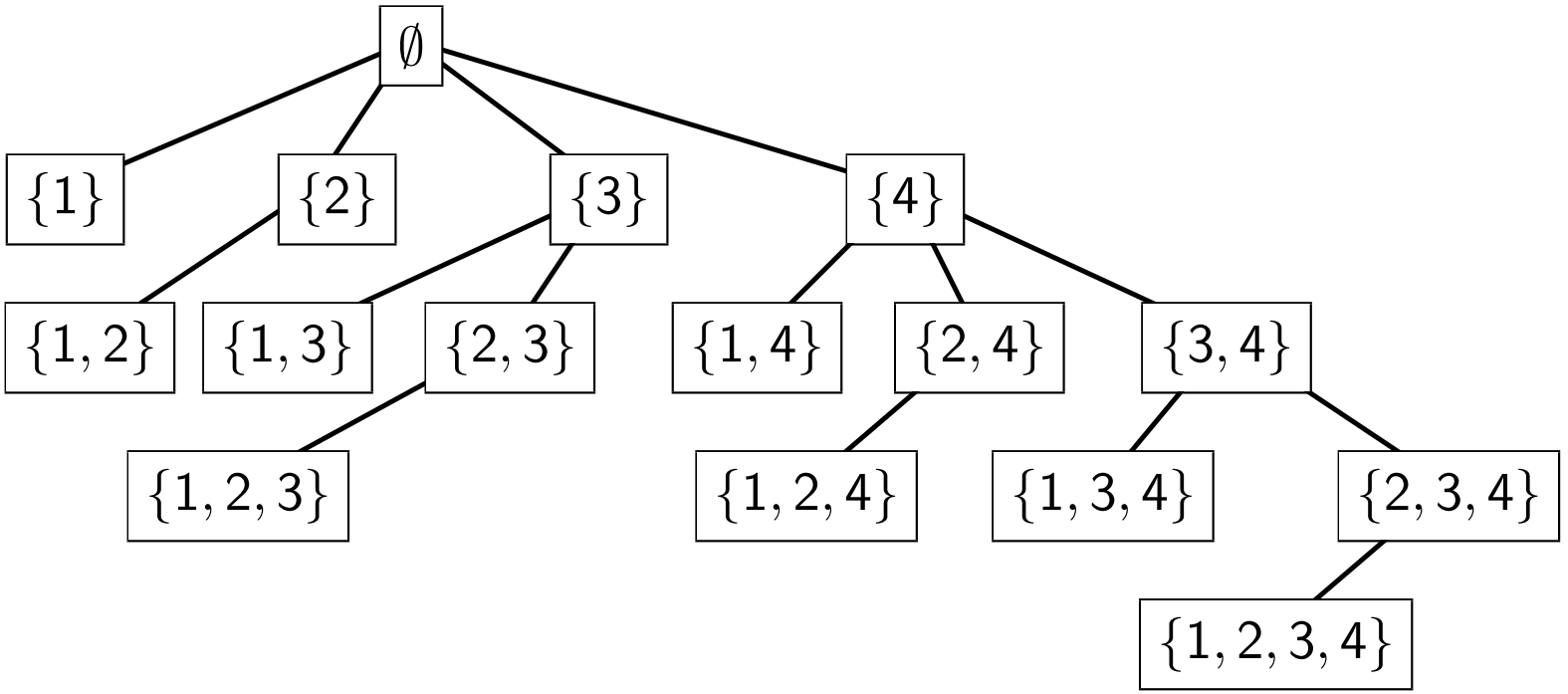
アルゴリズムの動作を見てみる

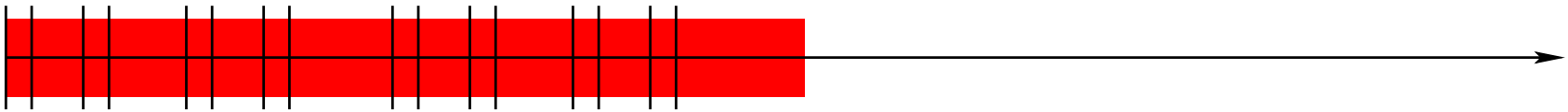




$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{2,3,4\}$, $\{1,2,3,4\}$

アルゴリズムの動作を見てみた





$\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{3\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{4\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{2,3,4\}$, $\{1,2,3,4\}$

部分集合列挙アルゴリズム

入力：自然数 n，出力：$\{1, \ldots, n\}$ の部分集合すべて

- $\mathrm{B}(\emptyset)$ **を呼び出し**

$\mathrm{B}(X)$

事前条件：$X \subseteq \{1, \ldots, n\}$
事後条件：列挙木における X と X の子孫をすべて出力

- X **を出力**
- $i := \min X$ **とする**
- $i = 1$ **ならば終了**
- **そうでなければ，** $j := 1$ **として，以下を繰り返し**
  - $j = i$ **または** $j > n$ **ならば終了**
  - **そうでなければ，** $\mathrm{B}(X \cup \{j\})$ **を呼び出し**
  - $j := j+1$

逆探索アルゴリズムの正当性と効率性

正当性

- $\mathrm{B}(X)$ が事後条件を満たすことは「親から子を得る操作」の正しさから得られる．

効率性：時間

- 再帰木の辺数 $= N - 1$ ($N =$ 出力の数 $= 2^n$)
- $\therefore$ 総計算時間 $= \mathrm{O}(N + nN) = \mathrm{O}(nN)$
- $\therefore$ ならし遅延 $= \mathrm{O}(n)$
- 再帰木の高さ $= n$
- $\therefore$ 最悪時遅延 $= \mathrm{O}(n)$
- (差分出力をしても差分の大きさが $\Omega(n)$ になりうるので，最悪時遅延のオーダーは減らない)

効率性：領域

- 再帰木の高さ $= n$ なので，領域計算量は $\mathrm{O}(n)$

前後順序探索 — 逆探索で効果的な差分出力をする工夫

前後順序探索 (prepostorder traversal) **で高速化**
(**偶奇探索** (odd-even traversal) **とも呼ぶ**)

前後順序探索の考え方

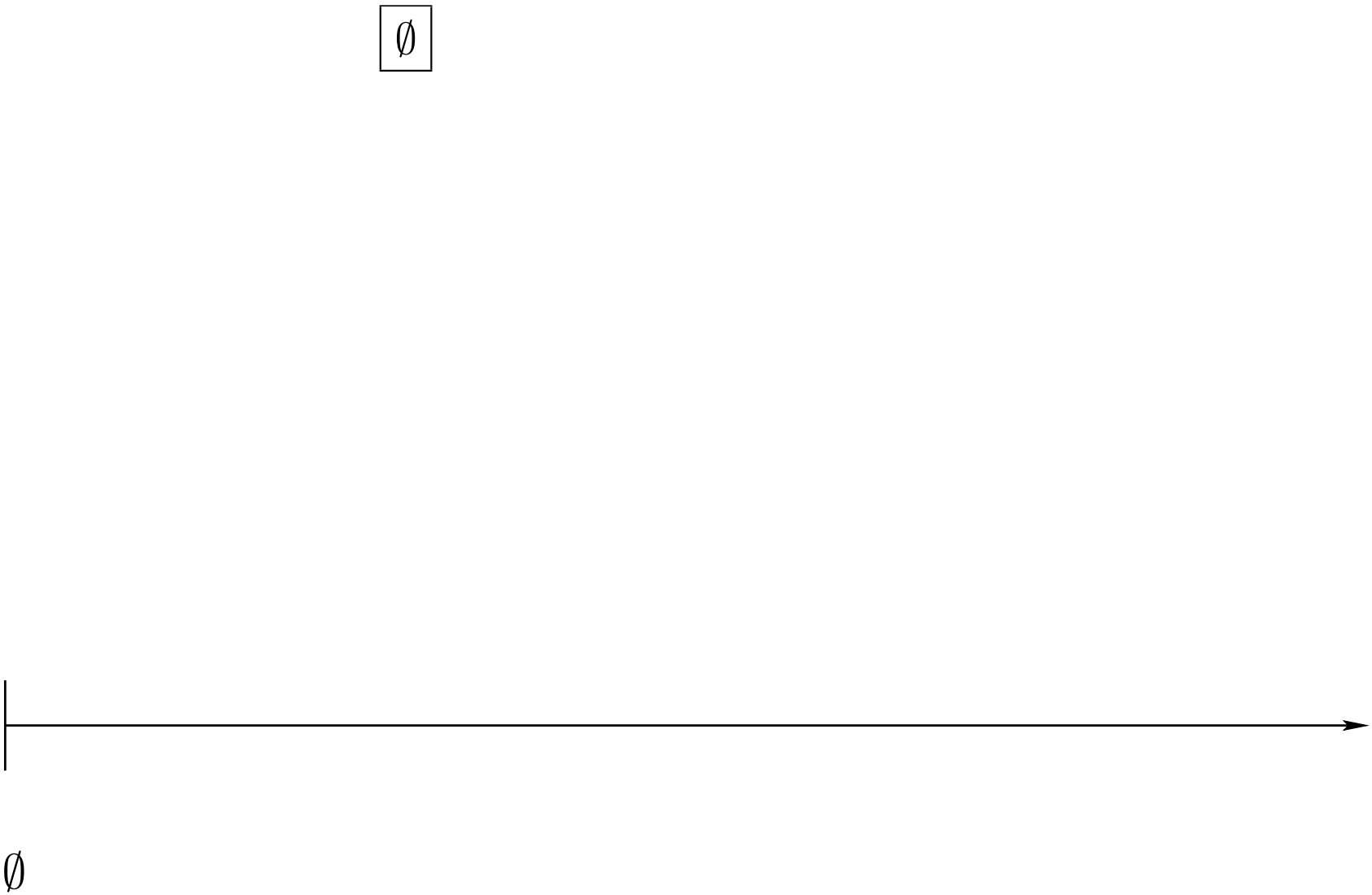
再帰木において

- **深さが偶数の頂点では，頂点を訪れたときに対応要素を出力**
  (自分を出力してから，子孫を出力)
- **深さが奇数の頂点では，頂点を離れるときに対応要素を出力**
  (子孫をすべて出力してから，自分を出力)

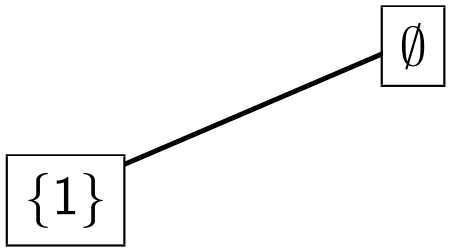
前後順序探索のメリット

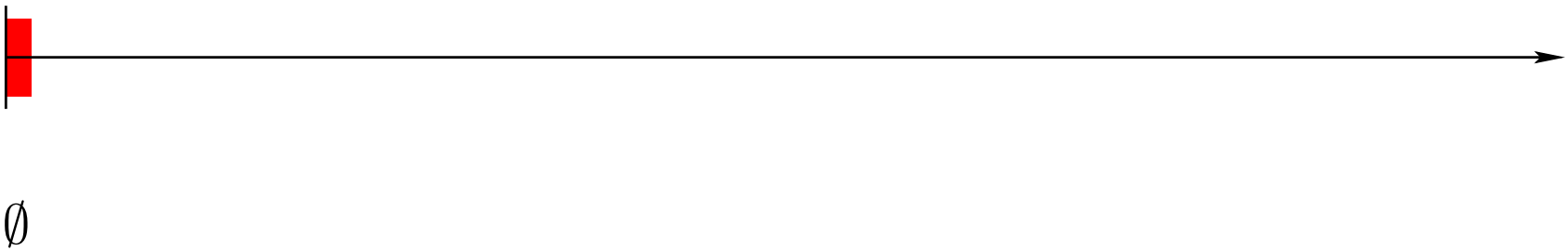
- **最悪時遅延が小さくなる**
- **出力の差分が小さくなる (定数となることが多い)**
- **∴ 前後順序探索と差分出力を組み合わせることで**
  「最悪時定数時間遅延」を達成できる (ことが多い)

前後順序探索の動きを見てみる

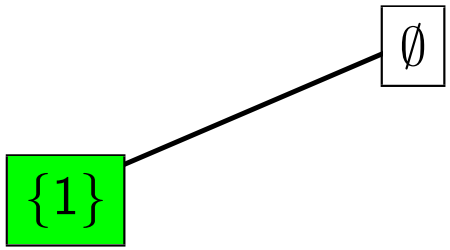


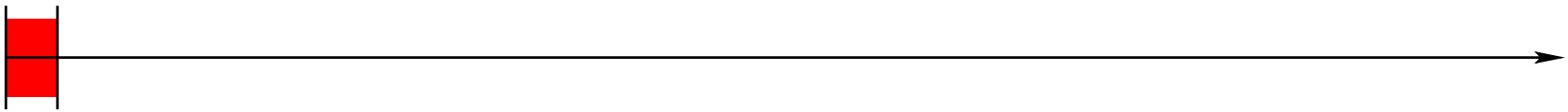
前後順序探索の動きを見てみる





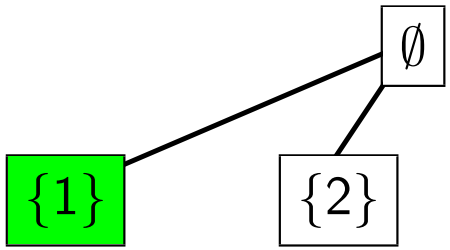
前後順序探索の動きを見てみる

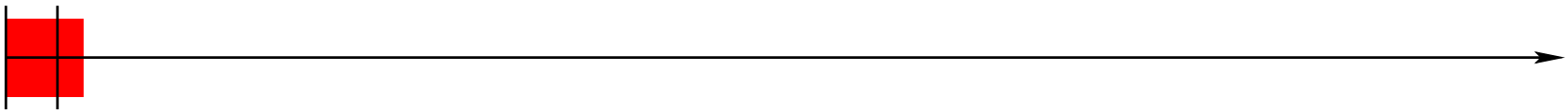




∅, {1}

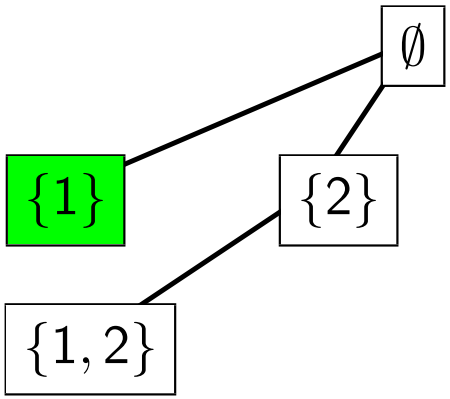
前後順序探索の動きを見てみる

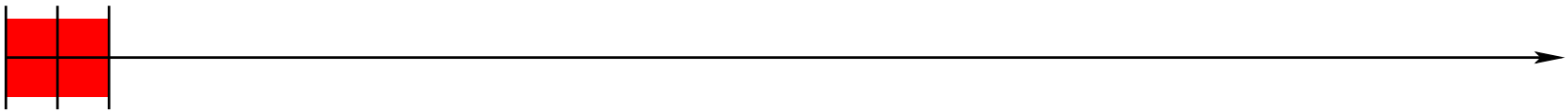




$\emptyset, \{1\}$

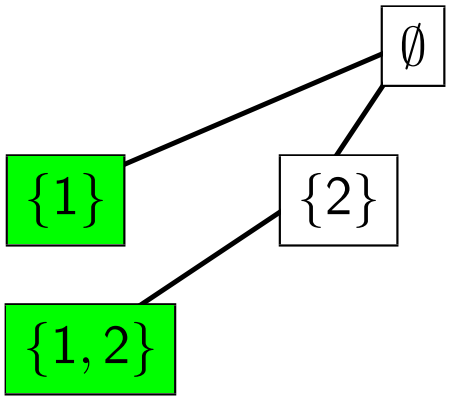
前後順序探索の動きを見てみる

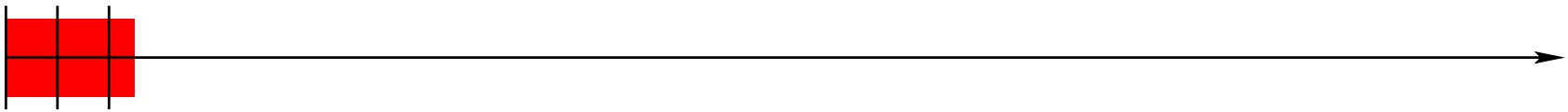




$\emptyset,\ \{1\},\ \{1,2\}$

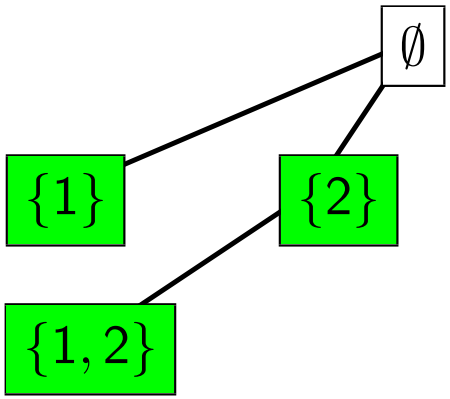
前後順序探索の動きを見てみる

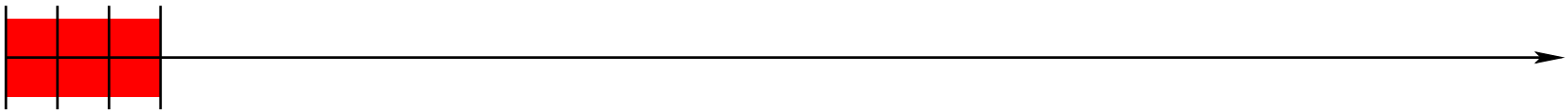




$\emptyset,\ \{1\},\ \{1,2\}$

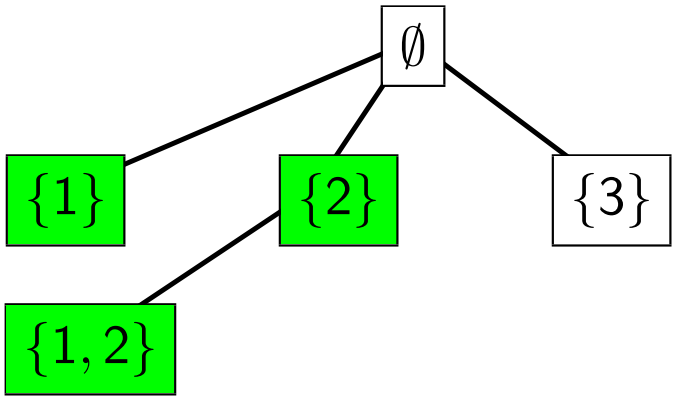
前後順序探索の動きを見てみる

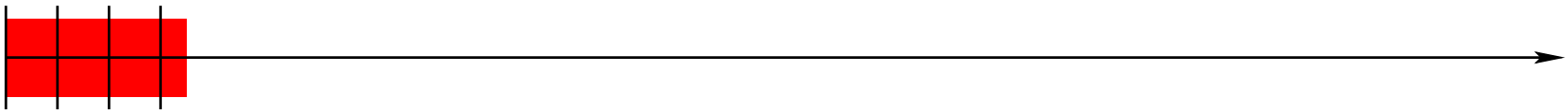




$\emptyset,\ \{1\},\ \{1,2\},\ \{2\}$

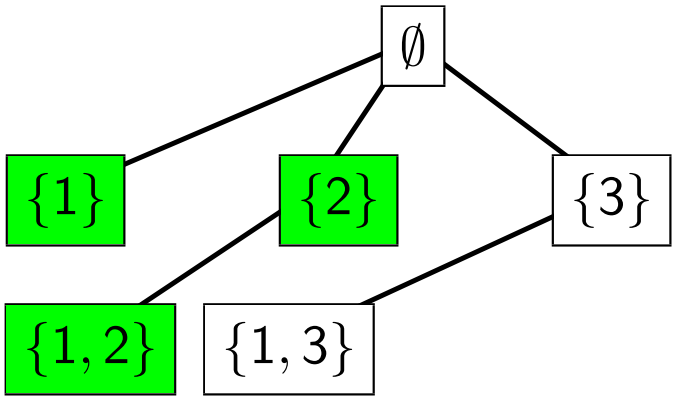
前後順序探索の動きを見てみる

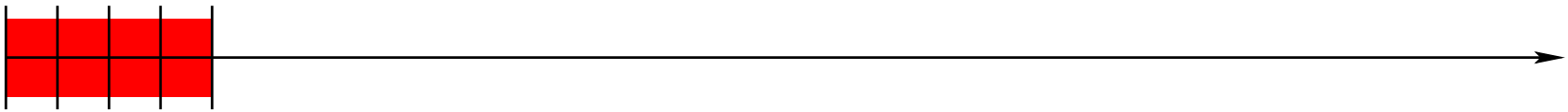




$\emptyset, \{1\}, \{1,2\}, \{2\}$

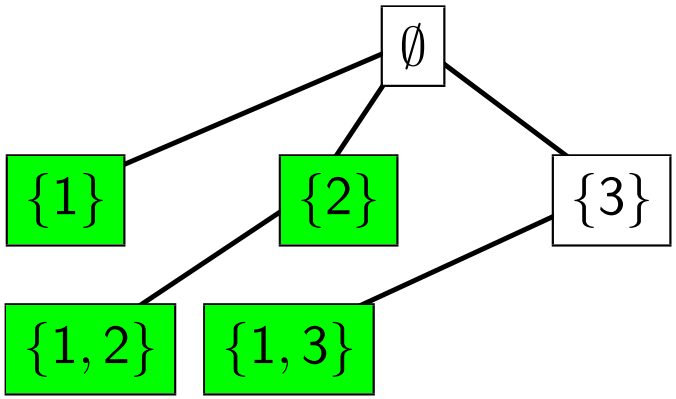
前後順序探索の動きを見てみる

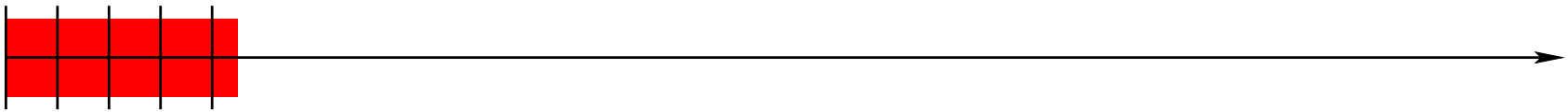




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$

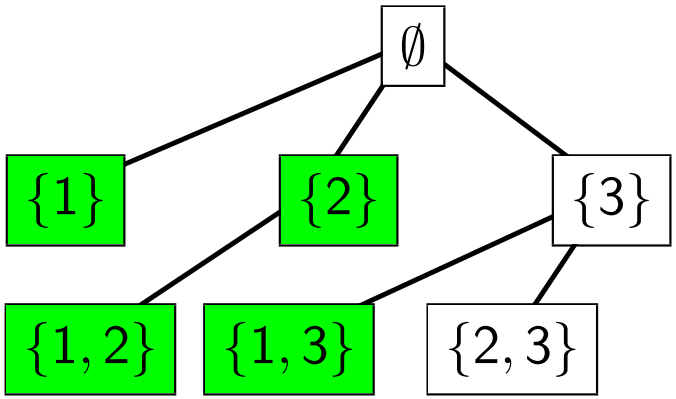
前後順序探索の動きを見てみる

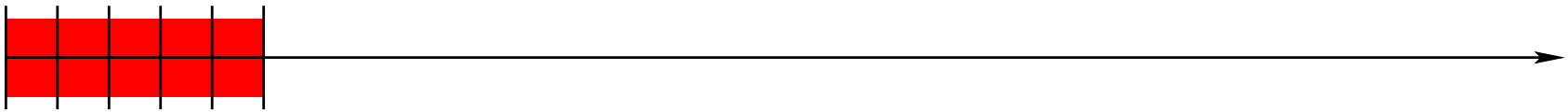




$\emptyset,\ \{1\},\ \{1,2\},\ \{2\},\ \{1,3\}$

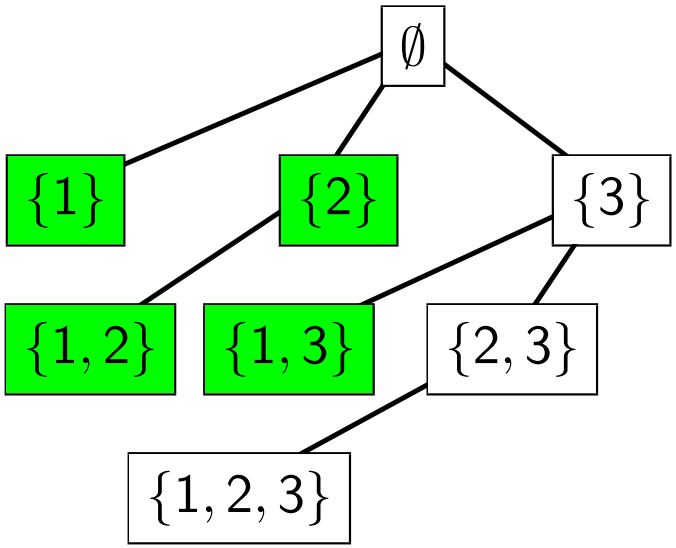
前後順序探索の動きを見てみる

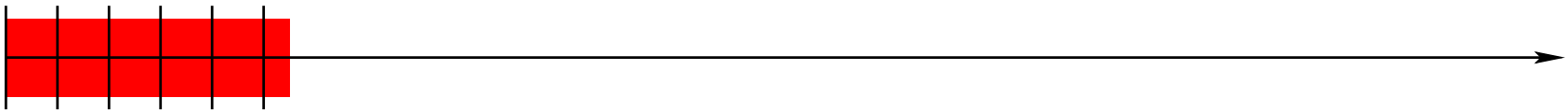




$\emptyset, \{1\}, \{1,2\}, \{2\}, \{1,3\}, \{2,3\}$

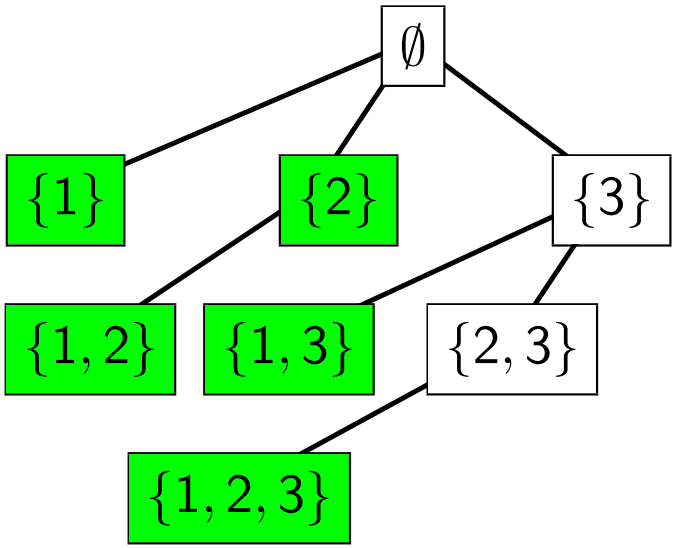
前後順序探索の動きを見てみる

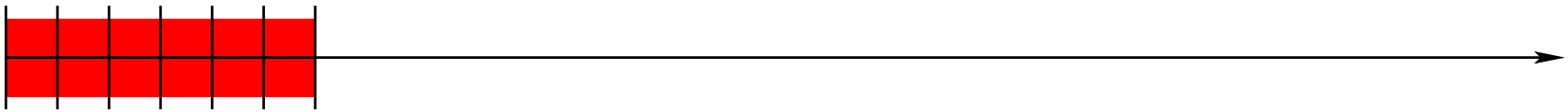




$\emptyset,\ \{1\},\ \{1,2\},\ \{2\},\ \{1,3\},\ \{2,3\}$

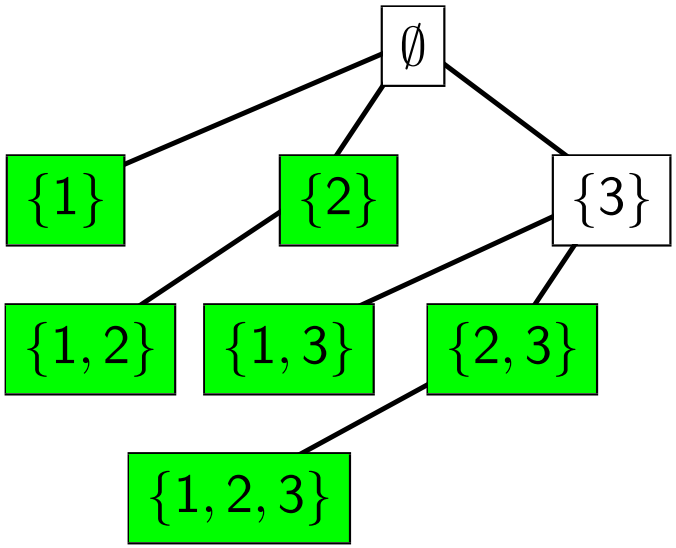
前後順序探索の動きを見てみる

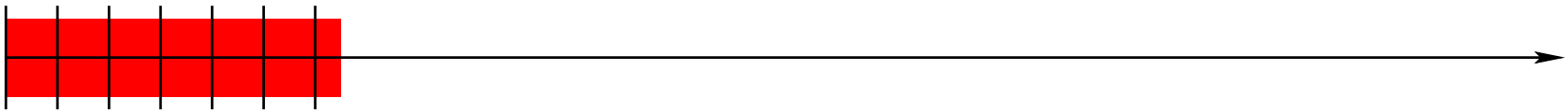




$\emptyset,\ \{1\},\ \{1,2\},\ \{2\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\}$

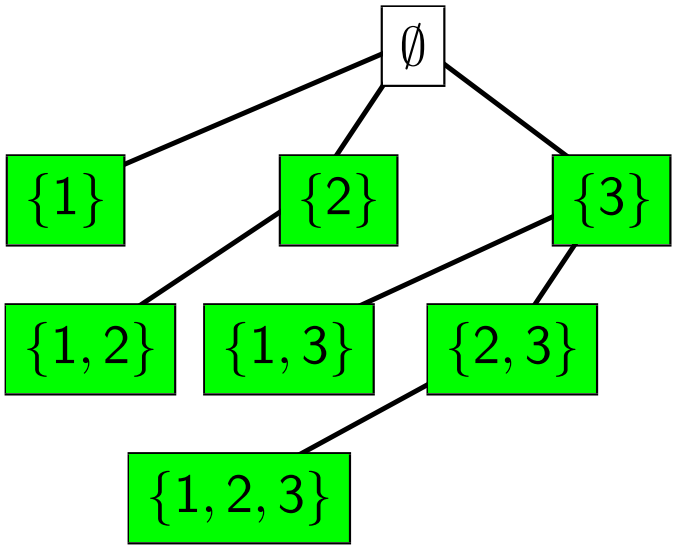
前後順序探索の動きを見てみる

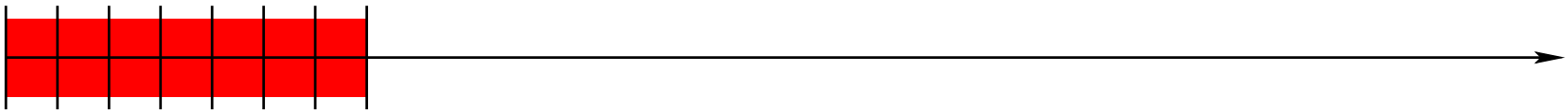




$\emptyset,\ \{1\},\ \{1,2\},\ \{2\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\}$

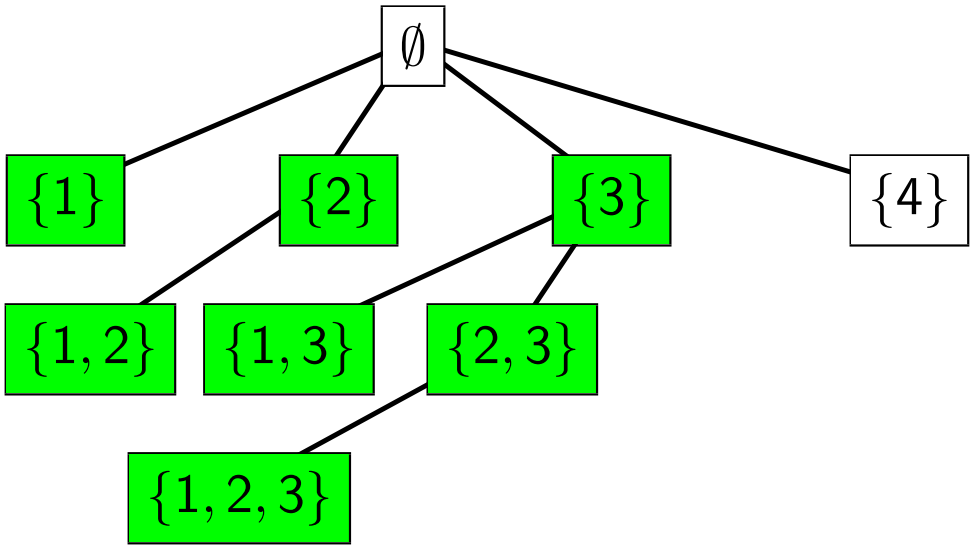
前後順序探索の動きを見てみる

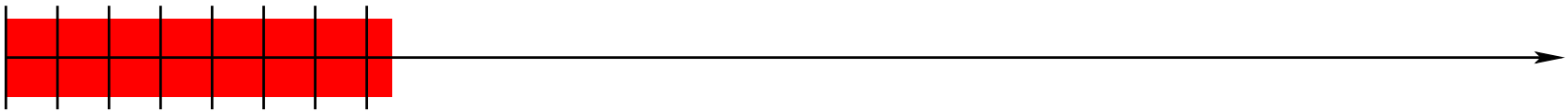




$\emptyset,\ \{1\},\ \{1,2\},\ \{2\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\},\ \{3\}$

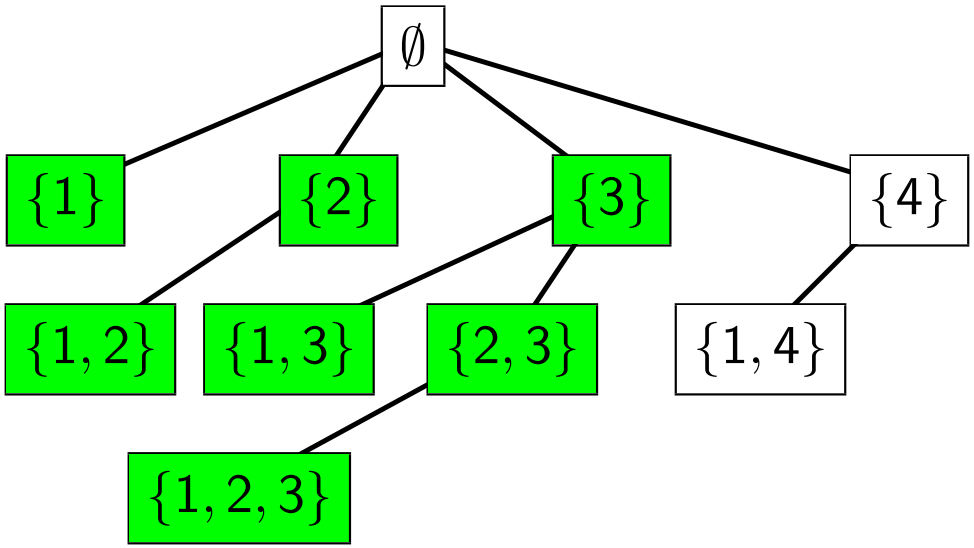
前後順序探索の動きを見てみる

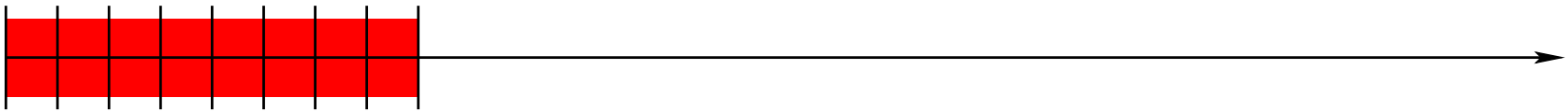




$\emptyset,\ \{1\},\ \{1,2\},\ \{2\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\},\ \{3\}$

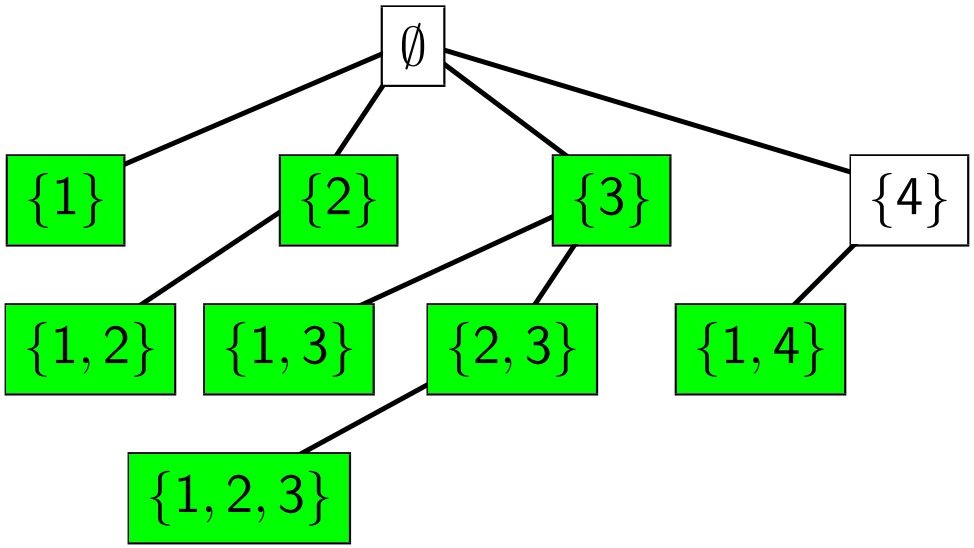
前後順序探索の動きを見てみる

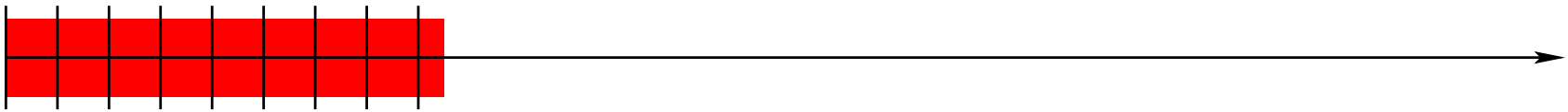




∅, {1}, {1, 2}, {2}, {1, 3}, {2, 3}, {1, 2, 3}, {3}, {1, 4}

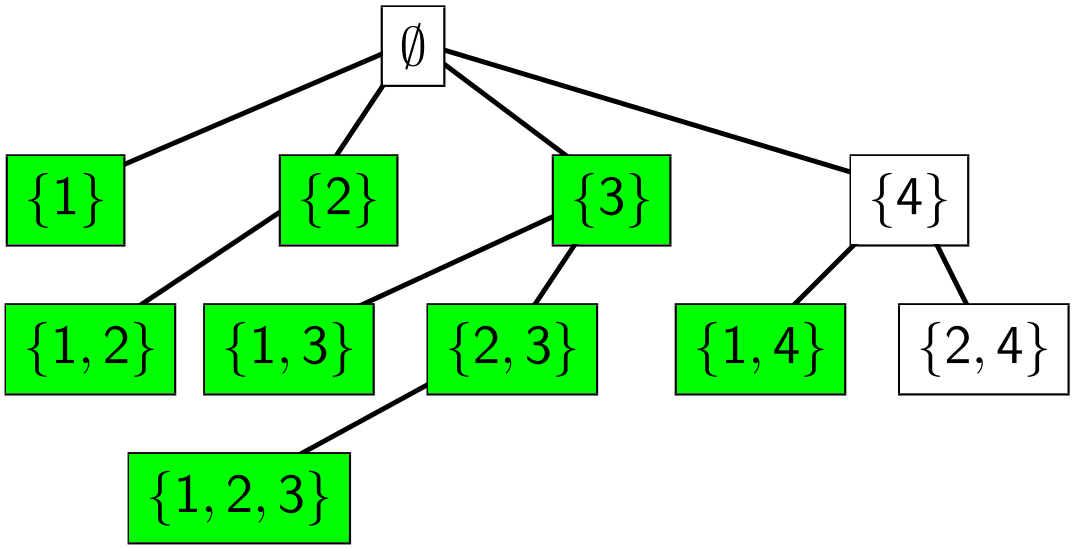
前後順序探索の動きを見てみる

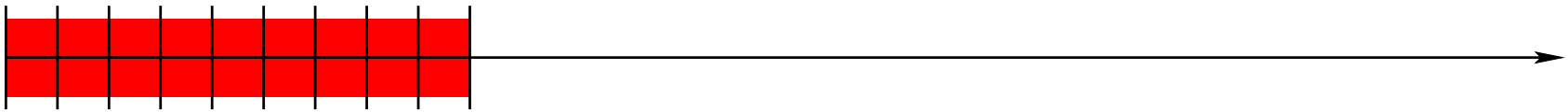




∅, {1}, {1, 2}, {2}, {1, 3}, {2, 3}, {1, 2, 3}, {3}, {1, 4}

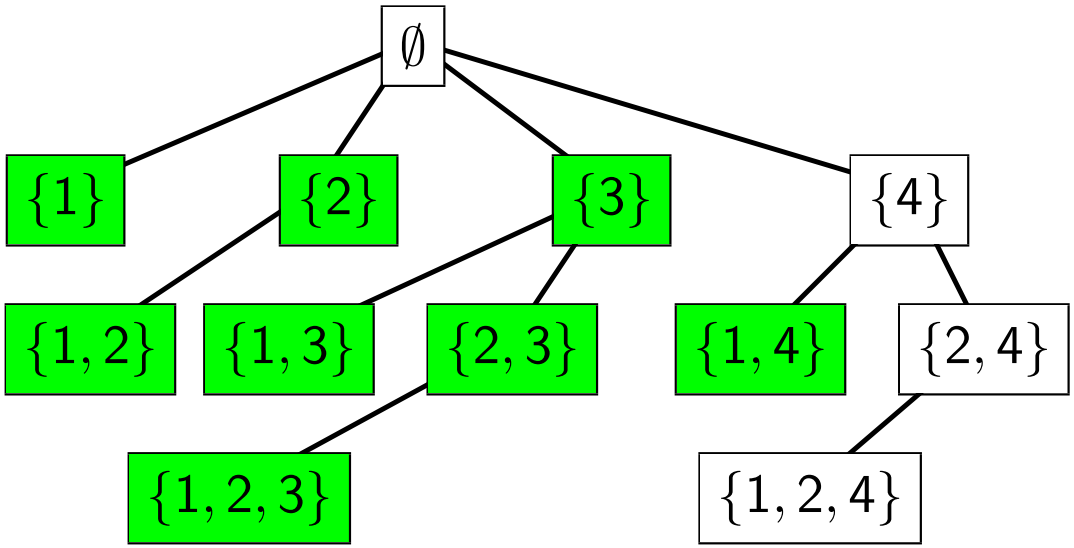
前後順序探索の動きを見てみる

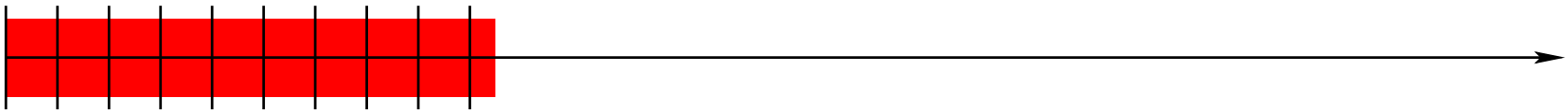




$\emptyset,\ \{1\},\ \{1,2\},\ \{2\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\},\ \{3\},\ \{1,4\},\ \{2,4\}$

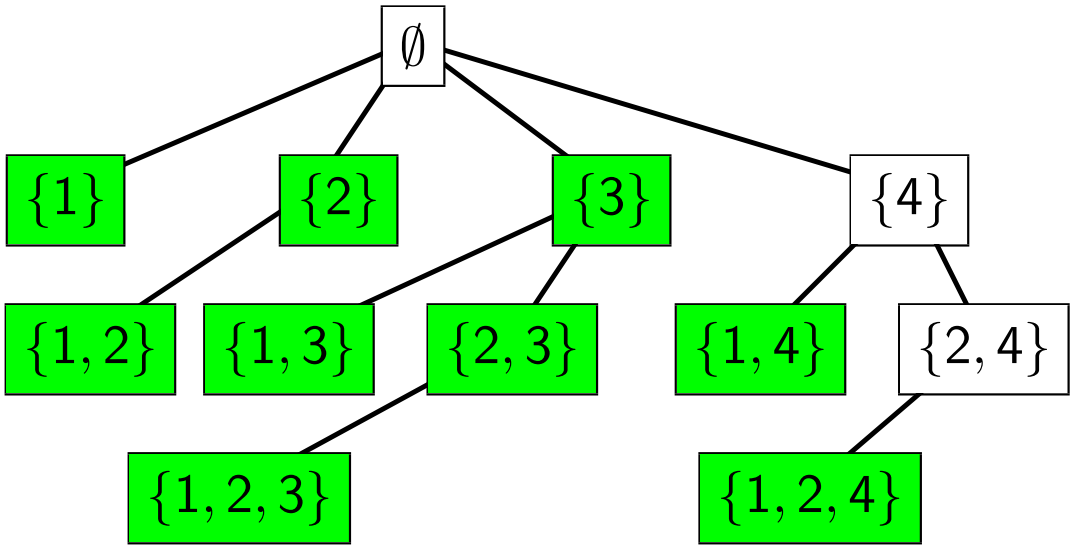
前後順序探索の動きを見てみる

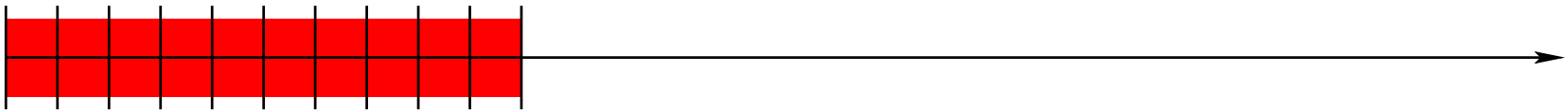




$\emptyset,\ \{1\},\ \{1,2\},\ \{2\},\ \{1,3\},\ \{2,3\},\ \{1,2,3\},\ \{3\},\ \{1,4\},\ \{2,4\}$

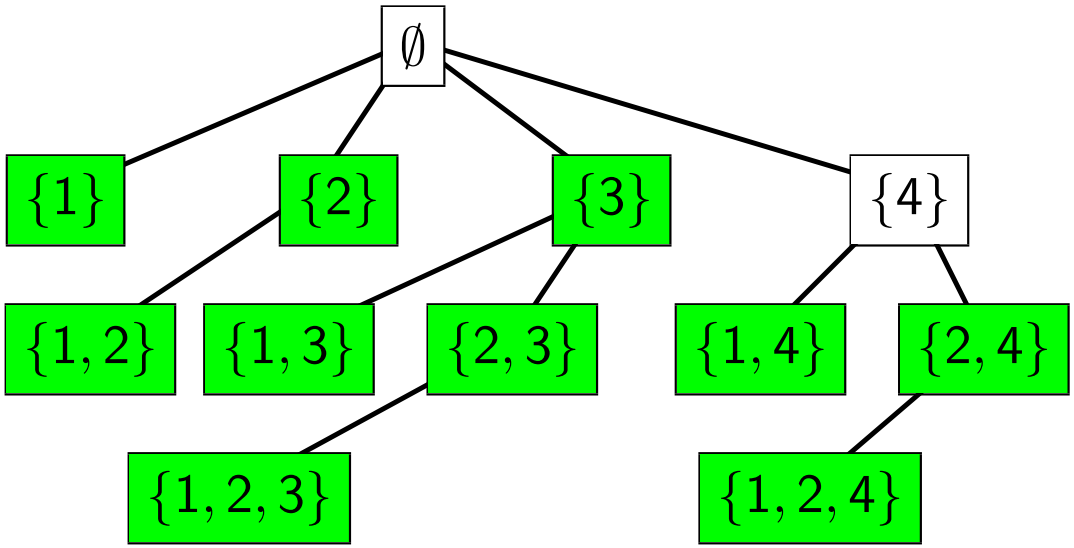
前後順序探索の動きを見てみる

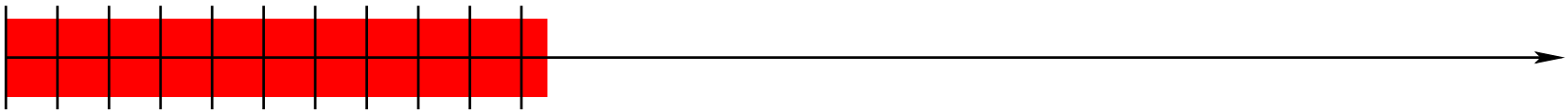




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$

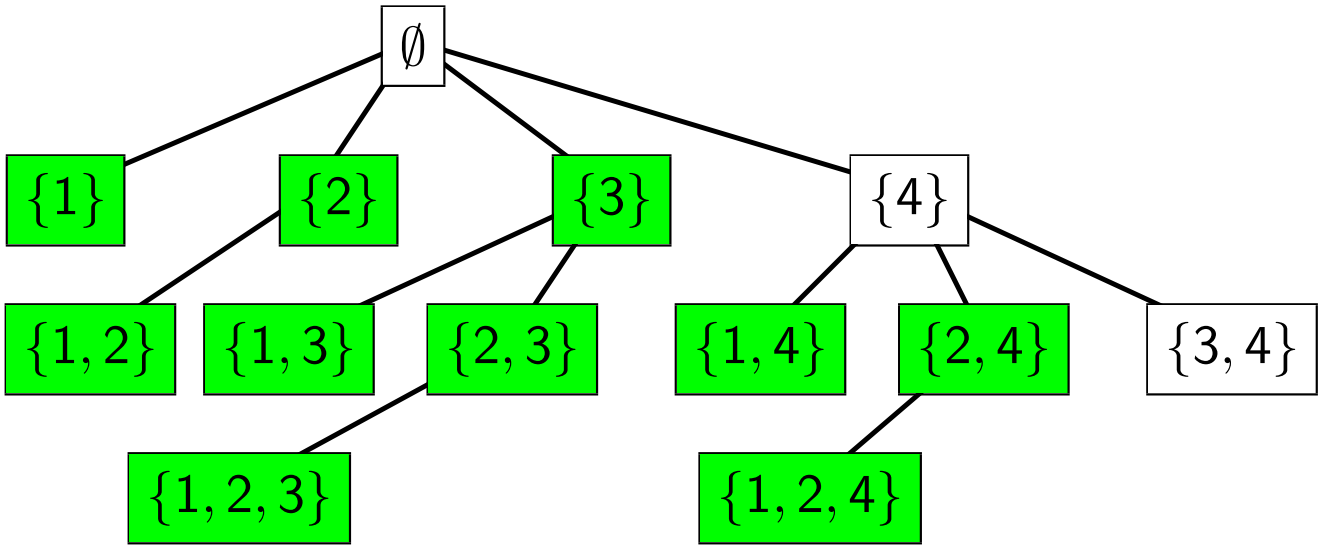
前後順序探索の動きを見てみる

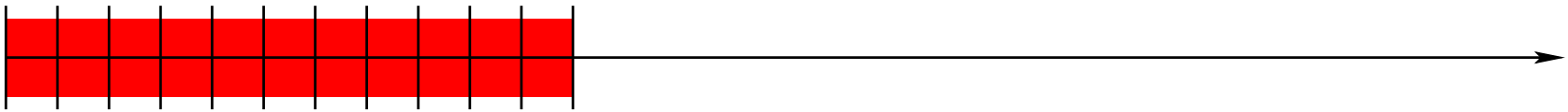




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$

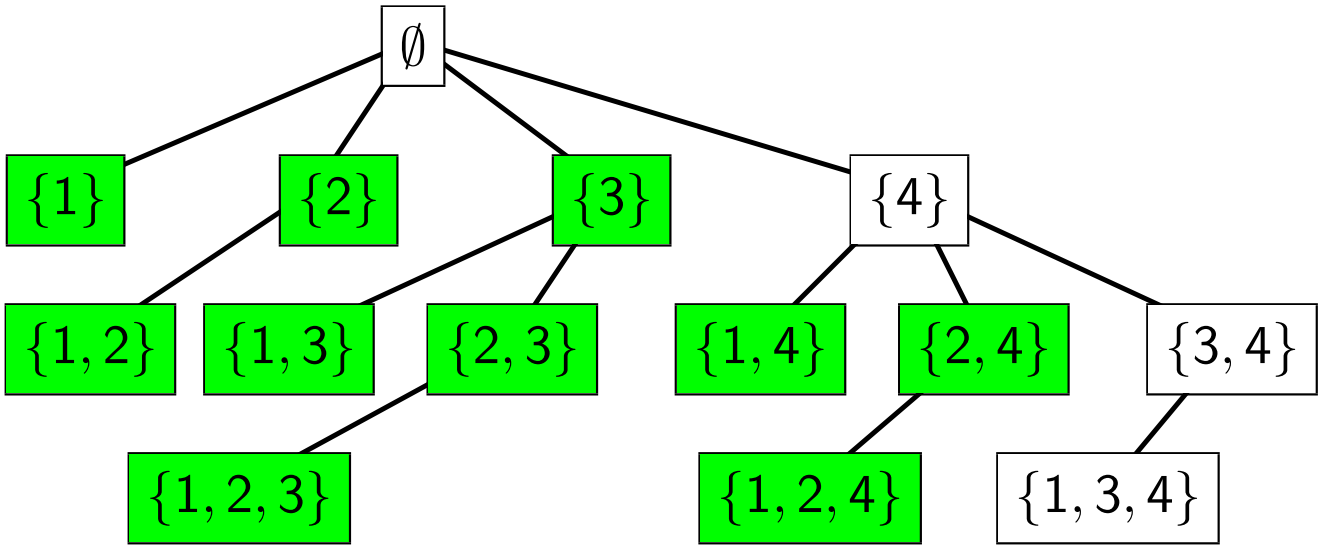
前後順序探索の動きを見てみる

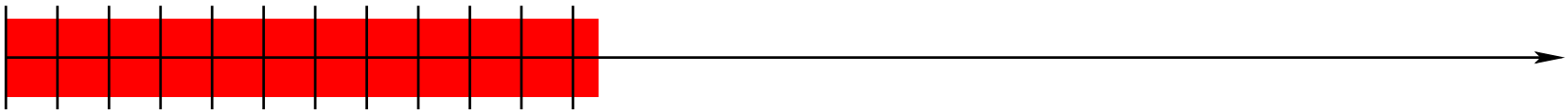




∅, {1}, {1, 2}, {2}, {1, 3}, {2, 3}, {1, 2, 3}, {3}, {1, 4}, {2, 4}, {1, 2, 4}, {3, 4}

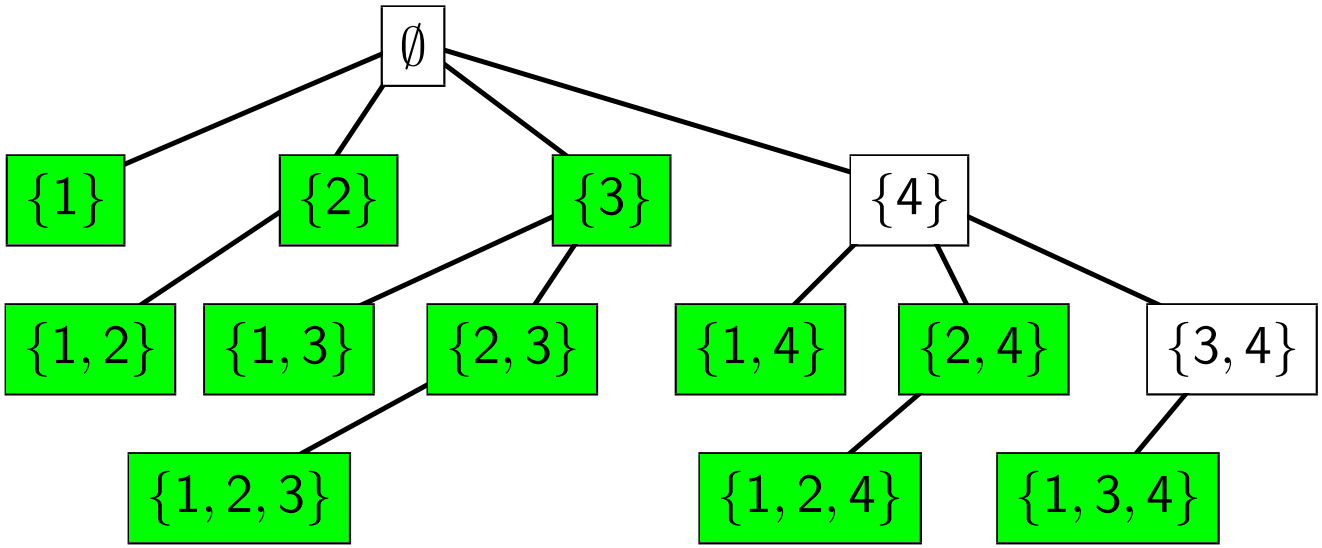
前後順序探索の動きを見てみる

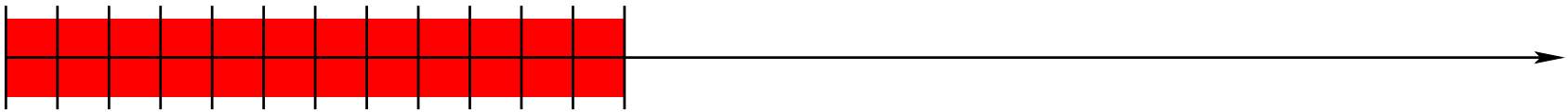




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$

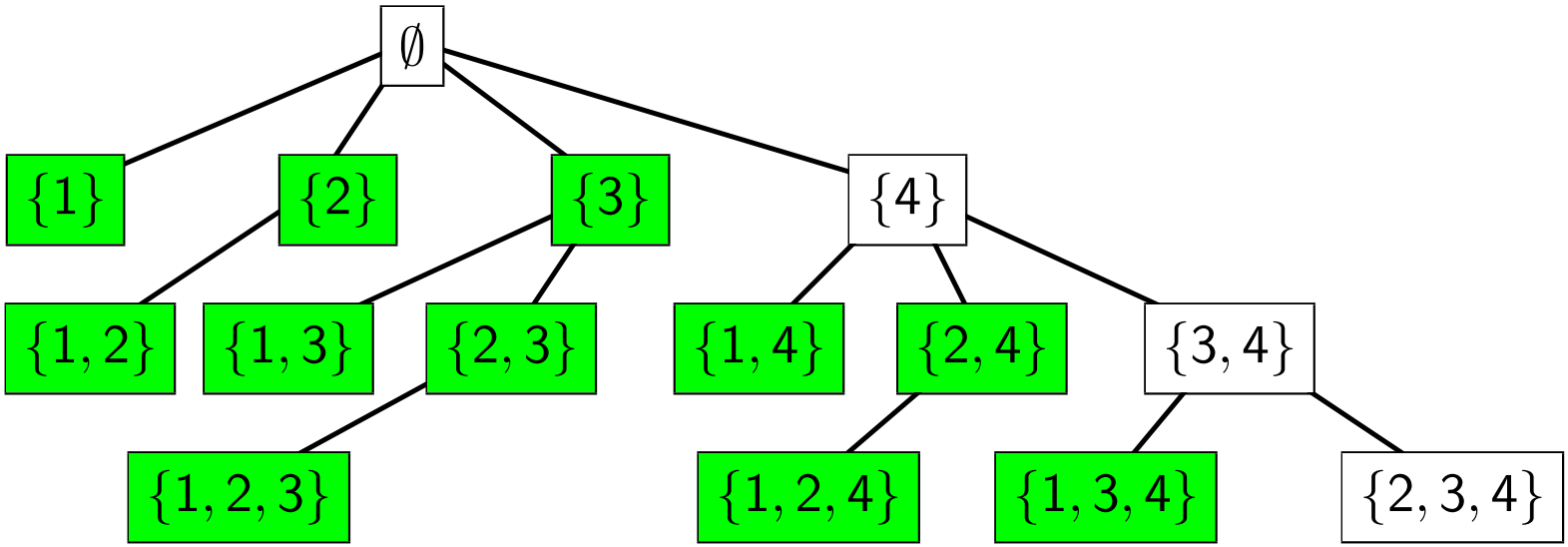
前後順序探索の動きを見てみる

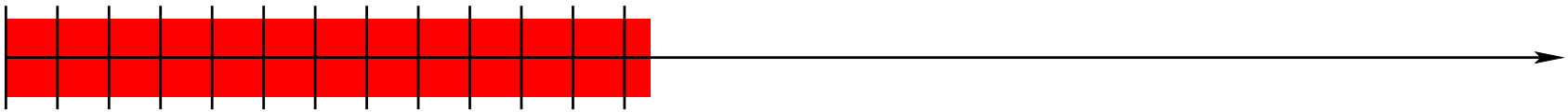




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$

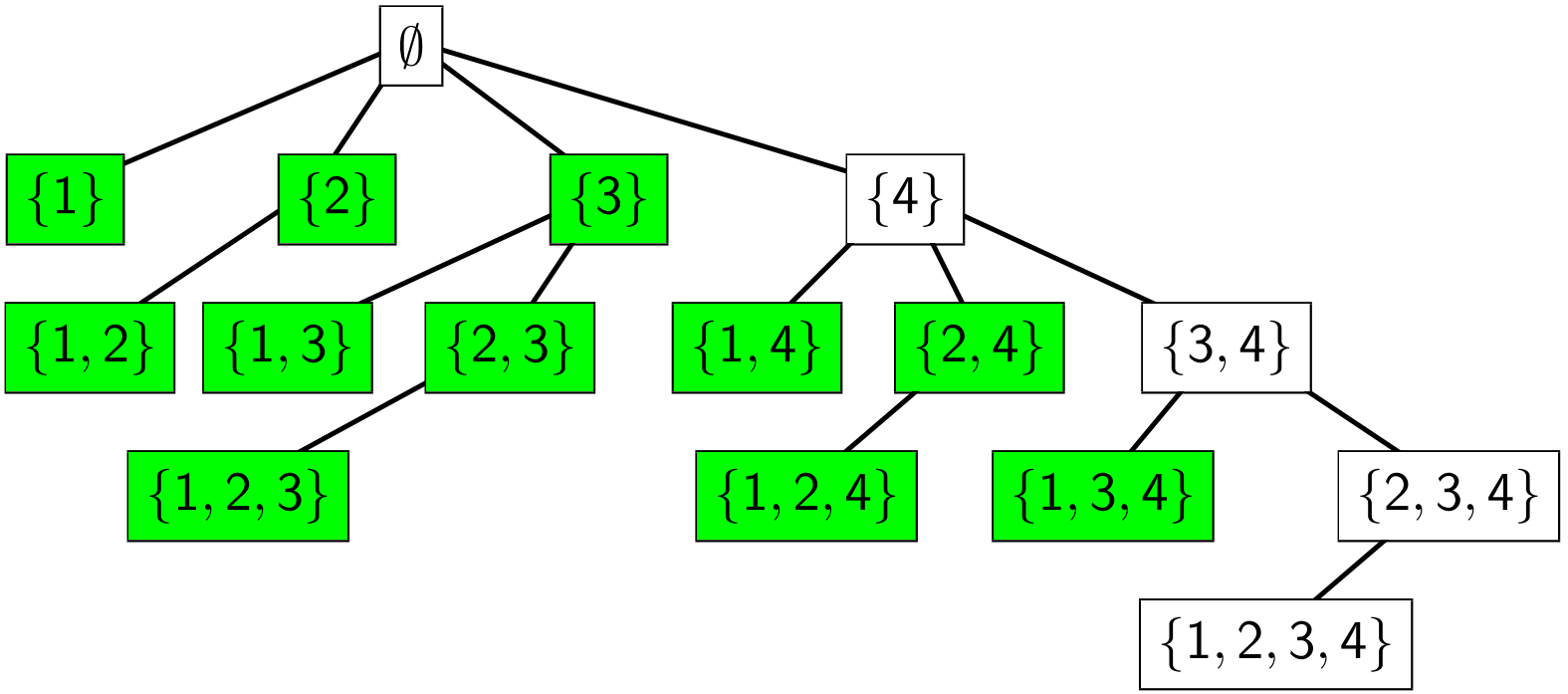
前後順序探索の動きを見てみる

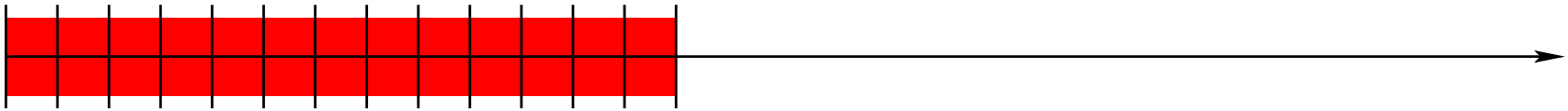




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$

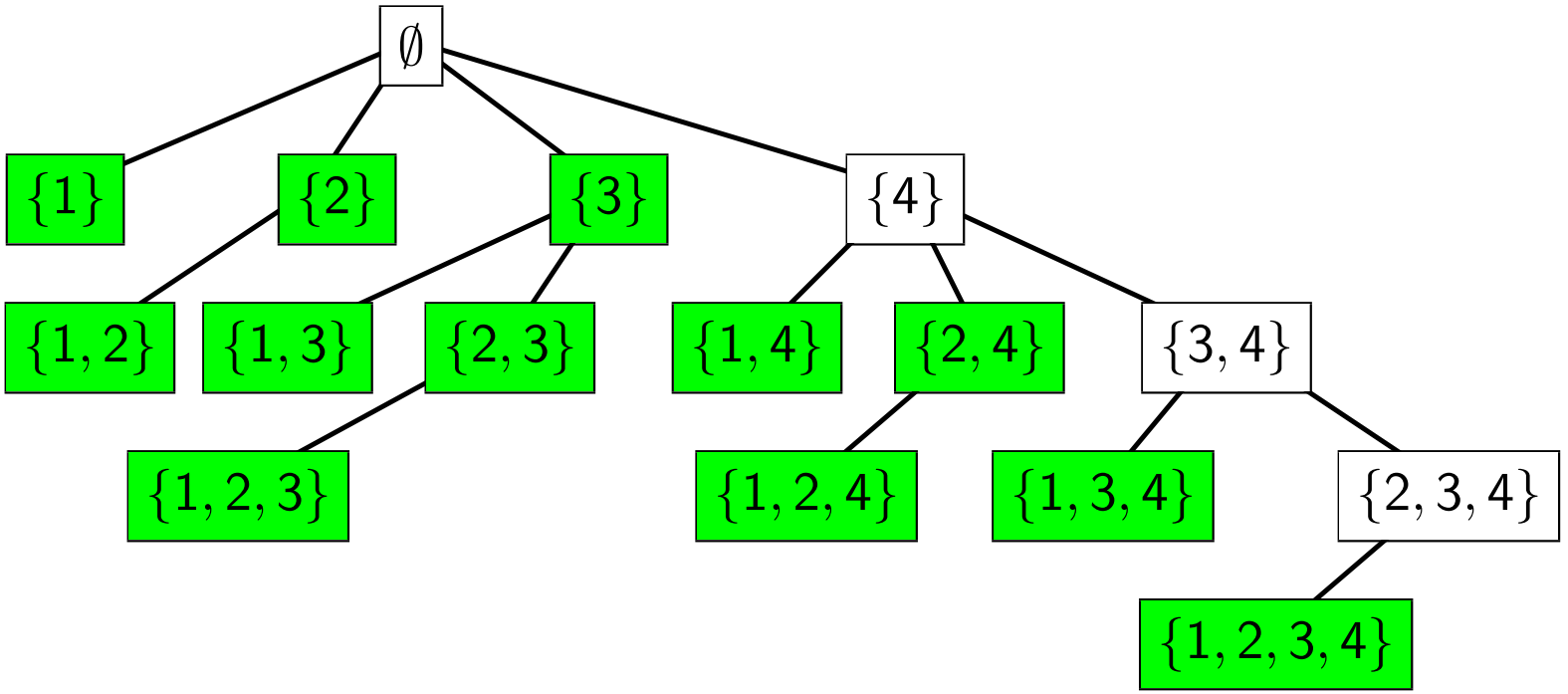
前後順序探索の動きを見てみる

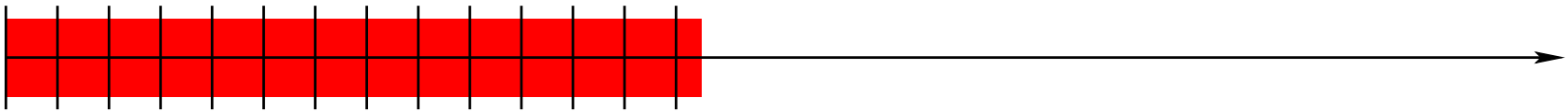




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{1,2,3,4\}$

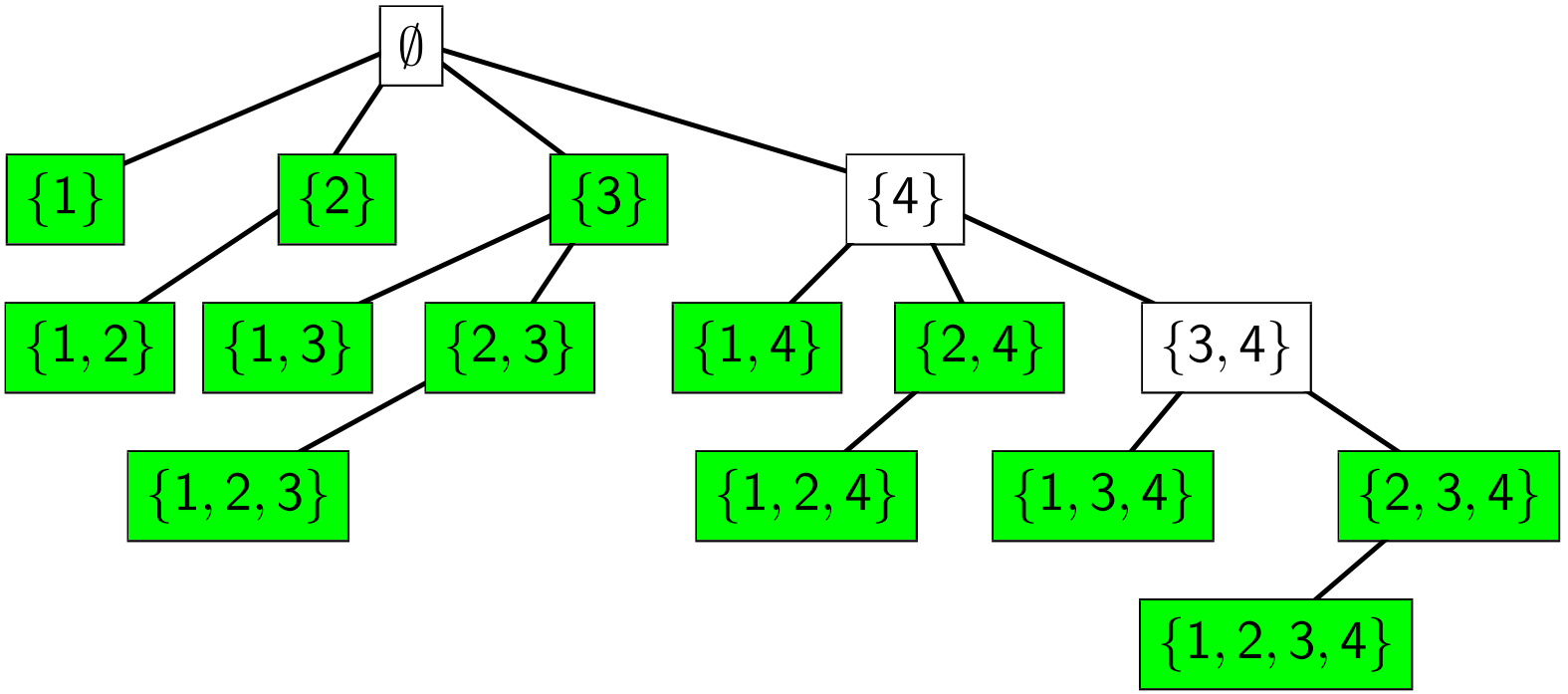
前後順序探索の動きを見てみる

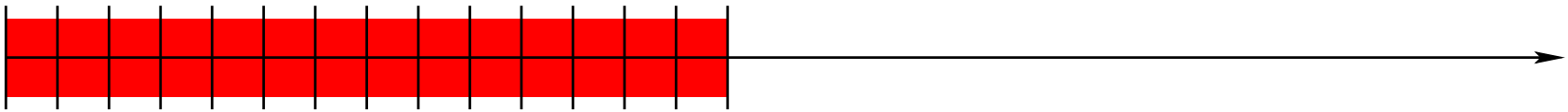




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{1,2,3,4\}$

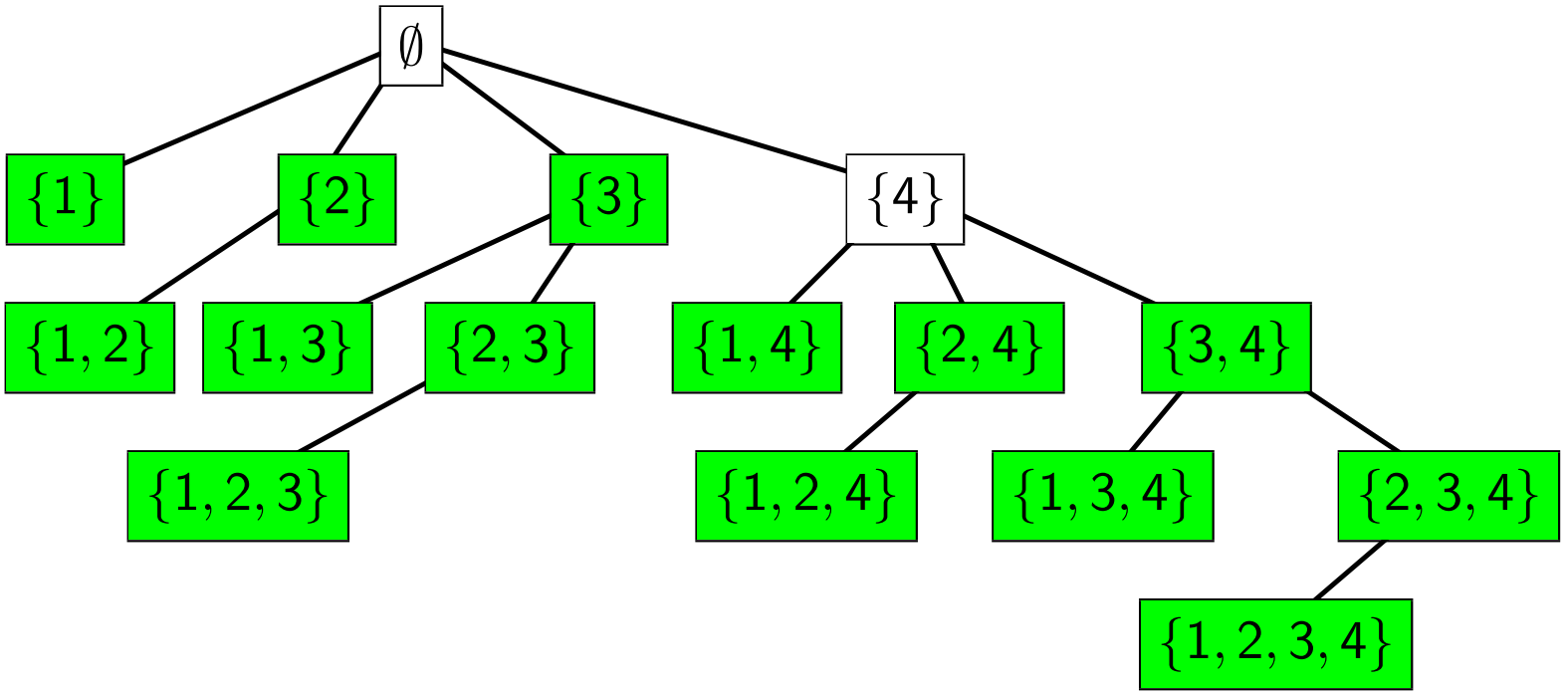
前後順序探索の動きを見てみる

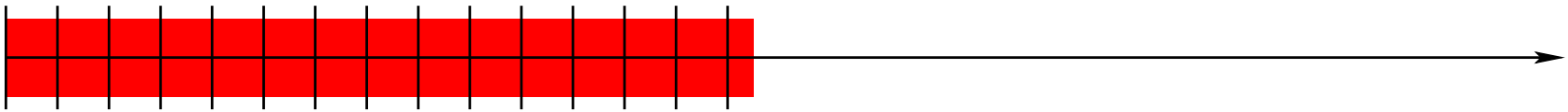




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{1,2,3,4\}$, $\{2,3,4\}$

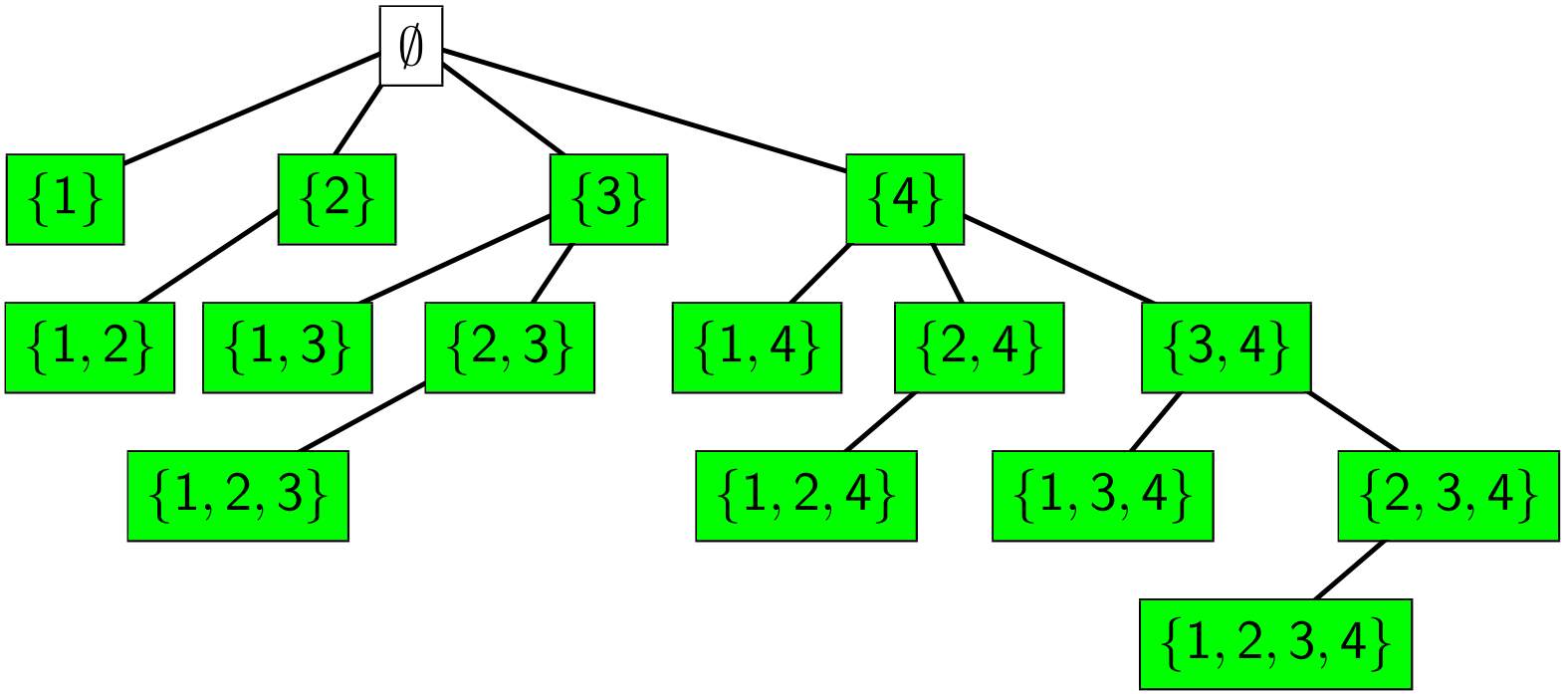
前後順序探索の動きを見てみる

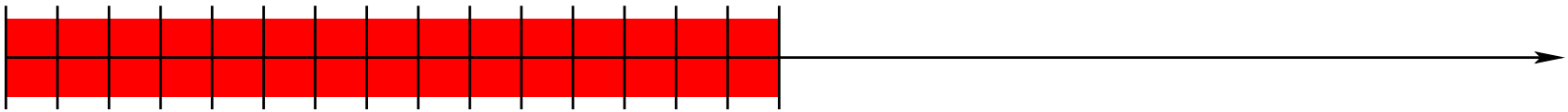




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{1,2,3,4\}$, $\{2,3,4\}$

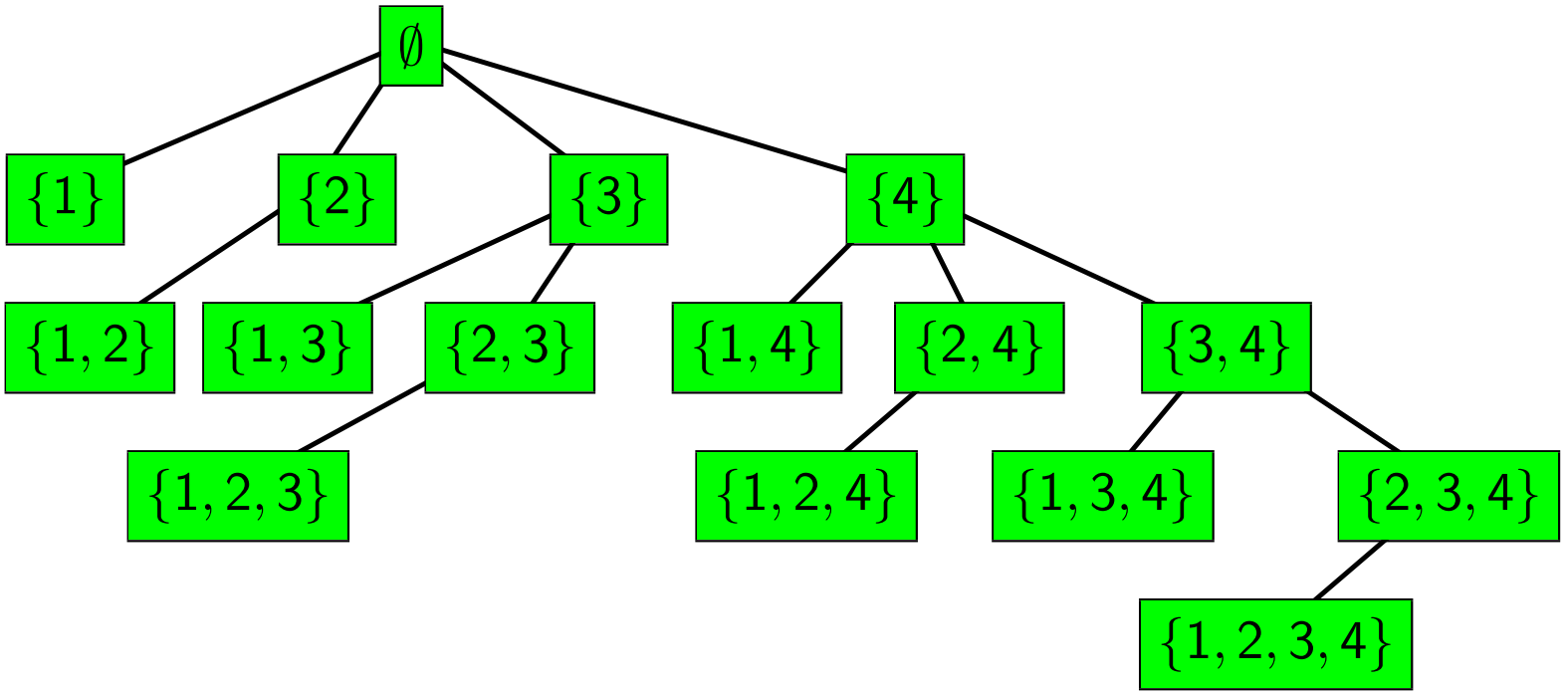
前後順序探索の動きを見てみる

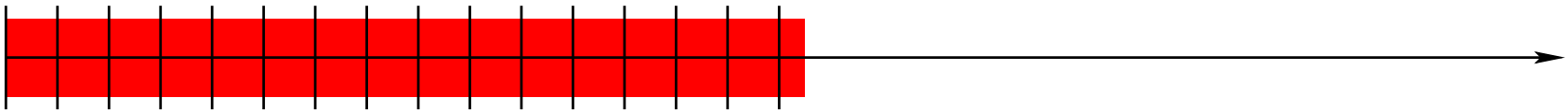




$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{1,2,3,4\}$, $\{2,3,4\}$, $\{4\}$

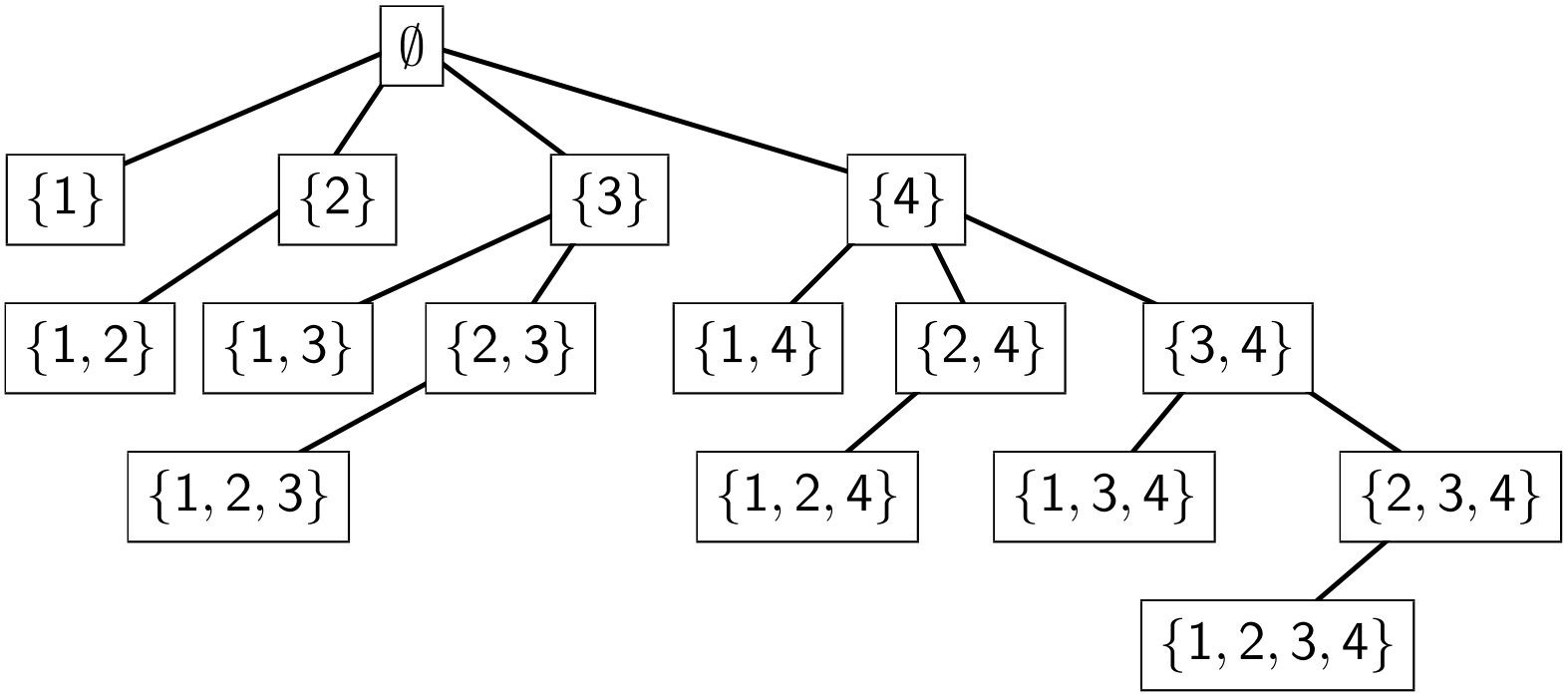
前後順序探索の動きを見てみる



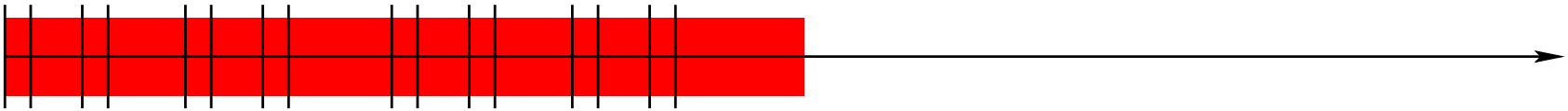


$\emptyset$, $\{1\}$, $\{1,2\}$, $\{2\}$, $\{1,3\}$, $\{2,3\}$, $\{1,2,3\}$, $\{3\}$, $\{1,4\}$, $\{2,4\}$, $\{1,2,4\}$, $\{3,4\}$, $\{1,3,4\}$, $\{1,2,3,4\}$, $\{2,3,4\}$, $\{4\}$

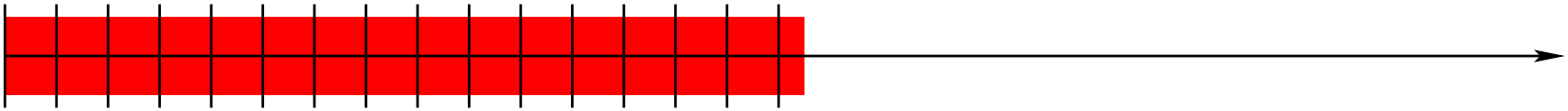
遅延の比較



普通の逆探索 (前順)



前後順序探索



差分出力の比較

普通の逆探索 (前順)

$\emptyset$,
$\{1\}$, $+1$,
$\{2\}$, $-1+2$,
$\{1,2\}$, $+1$,
$\{3\}$, $-1,2+3$,
$\{1,3\}$, $+1$,
$\{2,3\}$, $-1+2,3$,
$\{1,2,3\}$, $+1$,
$\{4\}$, $-1,2,3+4$,
$\{1,4\}$, $+1$,
$\{2,4\}$, $-1+2$,
$\{1,2,4\}$, $+1$,
$\{3,4\}$, $-1,2+3$,
$\{1,3,4\}$, $+1$,
$\{2,3,4\}$, $-1+2$,
$\{1,2,3,4\}$ $+1$

前後順序探索

$\emptyset$,
$\{1\}$, $+1$,
$\{1,2\}$, $+2$,
$\{2\}$, -1,
$\{1,3\}$, $-2+1,3$,
$\{2,3\}$, $-1+2$,
$\{1,2,3\}$, $+1$,
$\{3\}$, $-1,2$,
$\{1,4\}$, $-3+1,4$,
$\{2,4\}$, $-1+2$,
$\{1,2,4\}$, $+1$,
$\{3,4\}$, $-1,2+3$,
$\{1,3,4\}$, $+1$,
$\{1,2,3,4\}$, $+2$,
$\{2,3,4\}$, -1,
$\{4\}$ $-2,3$

部分集合列挙アルゴリズム

入力：自然数 n，出力：$\{1, \ldots, n\}$ の部分集合すべて

- B($\emptyset$,0) **を呼び出し**

B(X, p)

事前条件： $X \subseteq \{1, \ldots, n\}$, $p = |X| \mod 2$
事後条件：列挙木における X と X の子孫をすべて出力

- $p = 0$ **ならば** X **を出力**
- $i := \min X$ **とする**
- $i = 1$ **ならば次へ**
- **そうでなければ，** $j := 1$ **として，以下を繰り返し**
  - $j = i$ **または** $j > n$ **ならば繰り返しを抜けて次へ**
  - **そうでなければ，** B($X \cup \{j\}$, $p+1 \mod 2$) **を呼び出し**
  - $j := j+1$
- $p = 1$ **ならば** X **を出力**
- **終了**

前後順序探索における差分出力

- 前後順序探索は一般の根付き木の探索にも有効 (valid)

命題 4

任意の根付き木の上の前後順序探索において，任意の出力とその次の出力の間の (唯一の) 道の辺数はある定数以下

証明：演習問題
したがって

命題 5

部分集合列挙問題に対する逆探索アルゴリズム (+ 前後順序探索，差分出力) は最悪時定数時間遅延と多項式領域を達成する

パート 1 の内容

- **列挙問題，列挙アルゴリズムとは?**
- **列挙問題，列挙アルゴリズム設計の難しさ**
- **列挙アルゴリズム設計技法**
  - **分割探索** (binary partition)
  - **組合せ** Gray **符号** (combinatorial Gray code)
  - **逆探索** (reverse search)
- **難しい列挙問題**

手法の得失 (一般論)

分割探索

- − 再帰的な構造を持っている対象に有効
- + 設計しやすい
- − 遅延があまり小さくならない (特に大きいわけでもないが)

組合せ Gray 符号

- − 単純な構造を持っている対象に有効
- + 遅延が小さい (差分出力を併用)
- − 設計しにくい (ハミルトン道が存在するか?)
- − 「最悪時定数時間遅延」になりにくい

逆探索

- + 少々複雑な構造を持っている対象にも有効
- + 遅延が小さい (差分出力，前後順序探索を併用)
- + 「最悪時定数時間遅延」にしやすい
- − 設計に「慣れ」が必要

例をもう 1 つ

部分集合列挙問題 (済)

入力：自然数 n
出力：集合 $\{1, 2, \ldots, n\}$ の部分集合すべて

順列列挙問題

入力：自然数 n
出力：集合 $\{1, 2, \ldots, n\}$ の順列すべて

この講義での順列

定義

集合 $X \subseteq \{1, \ldots, n\}$ の順列 (permutation) を数列

$$(a_1, \ldots, a_m)$$

で次を満たすものとする (ただし，$|X| = m$)

- **任意の $e \in X$ に対して $e = a_i$ となる唯一の $i \in \{1, \ldots, m\}$ が存在する**

- **例：$(2, 4, 3, 6)$ は $\{2, 3, 4, 6\}$ の順列**
- **簡単のために「2436」「$2, 4, 3, 6$」とも書くことにする**
- **空列は ε で表すことにする**

順列列挙問題へのアプローチ

分割探索によるアプローチ

- むしろ「バックトラック」と呼ぶものかもしれない

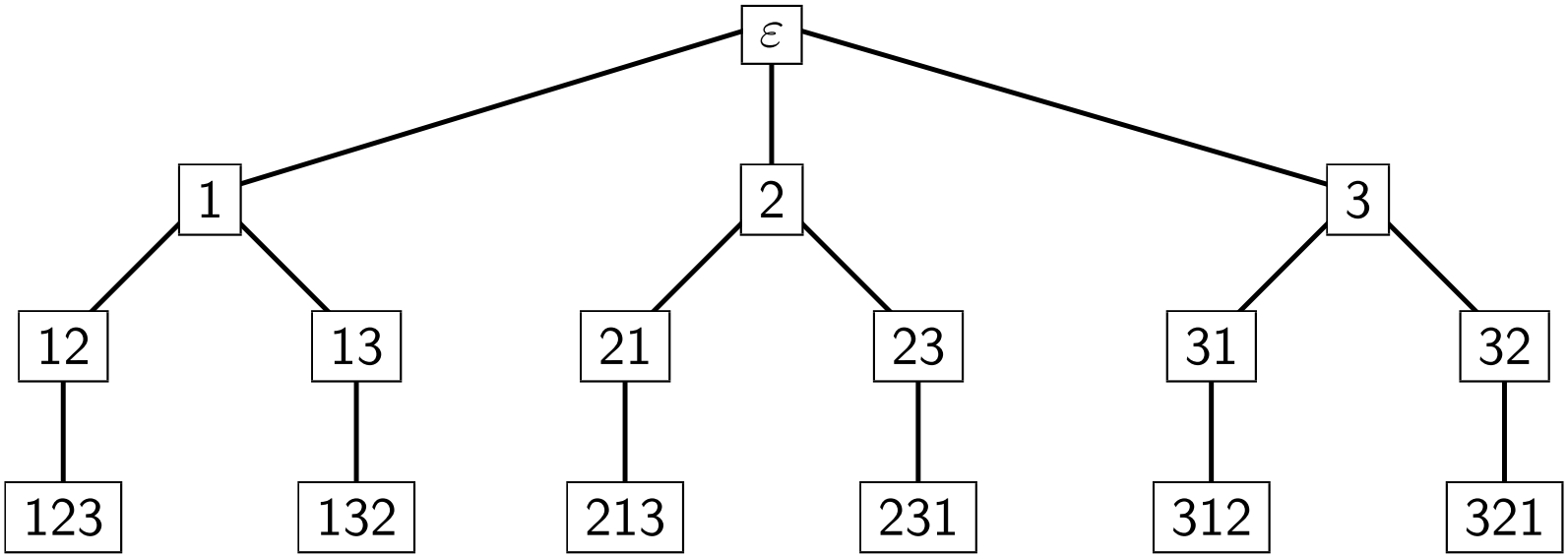
組合せ Gray 符号によるアプローチ

- グラフの定義，ハミルトン道の存在性
- ハミルトン道上の探索法

逆探索によるアプローチ

- 根付き木，親子関係の定義
- 親から子を見つける方法

バックトラッキングによる順列列挙アルゴリズム (の動き)



順列列挙問題へのアプローチ

分割探索によるアプローチ

- むしろ「バックトラック」と呼ぶものかもしれない

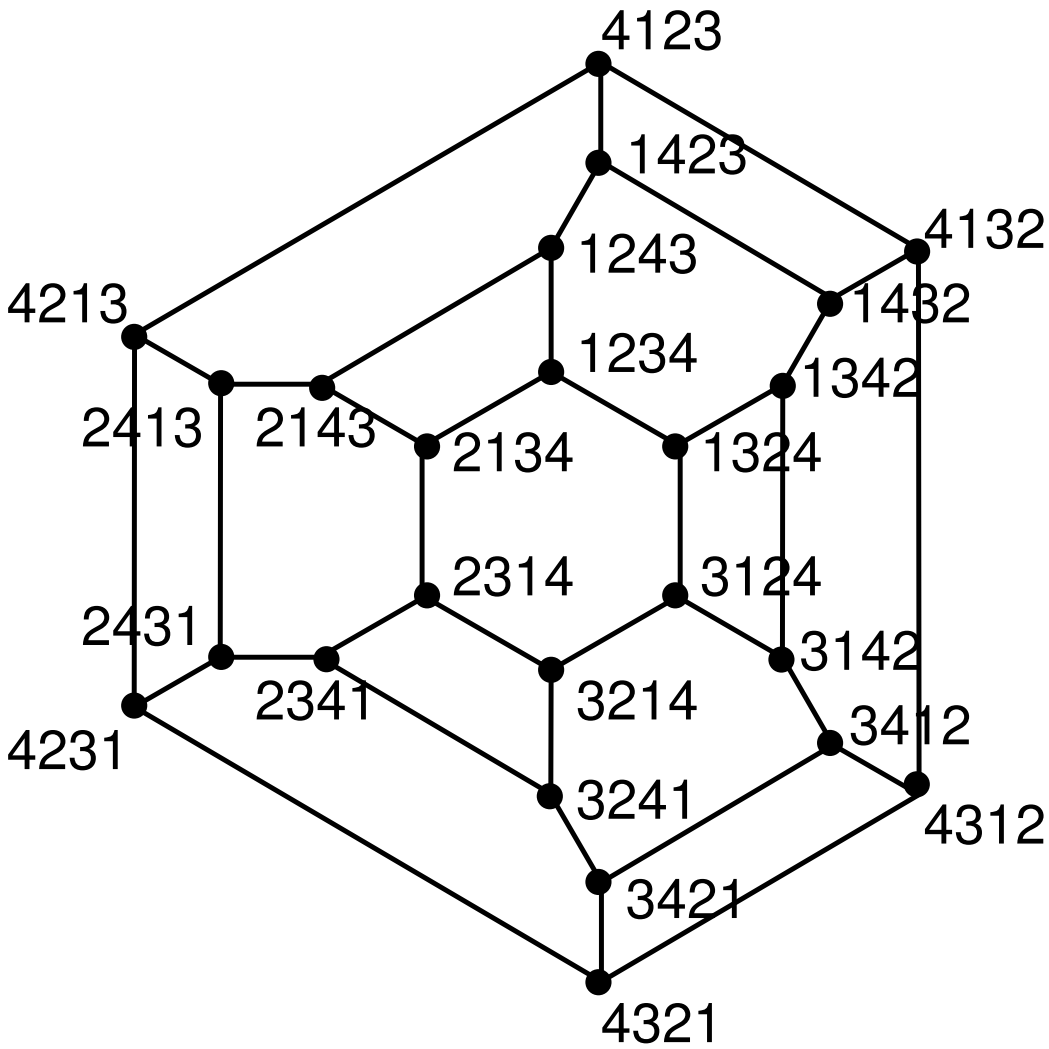
組合せ Gray 符号によるアプローチ

- グラフの定義，ハミルトン道の存在性
- ハミルトン道上の探索法

逆探索によるアプローチ

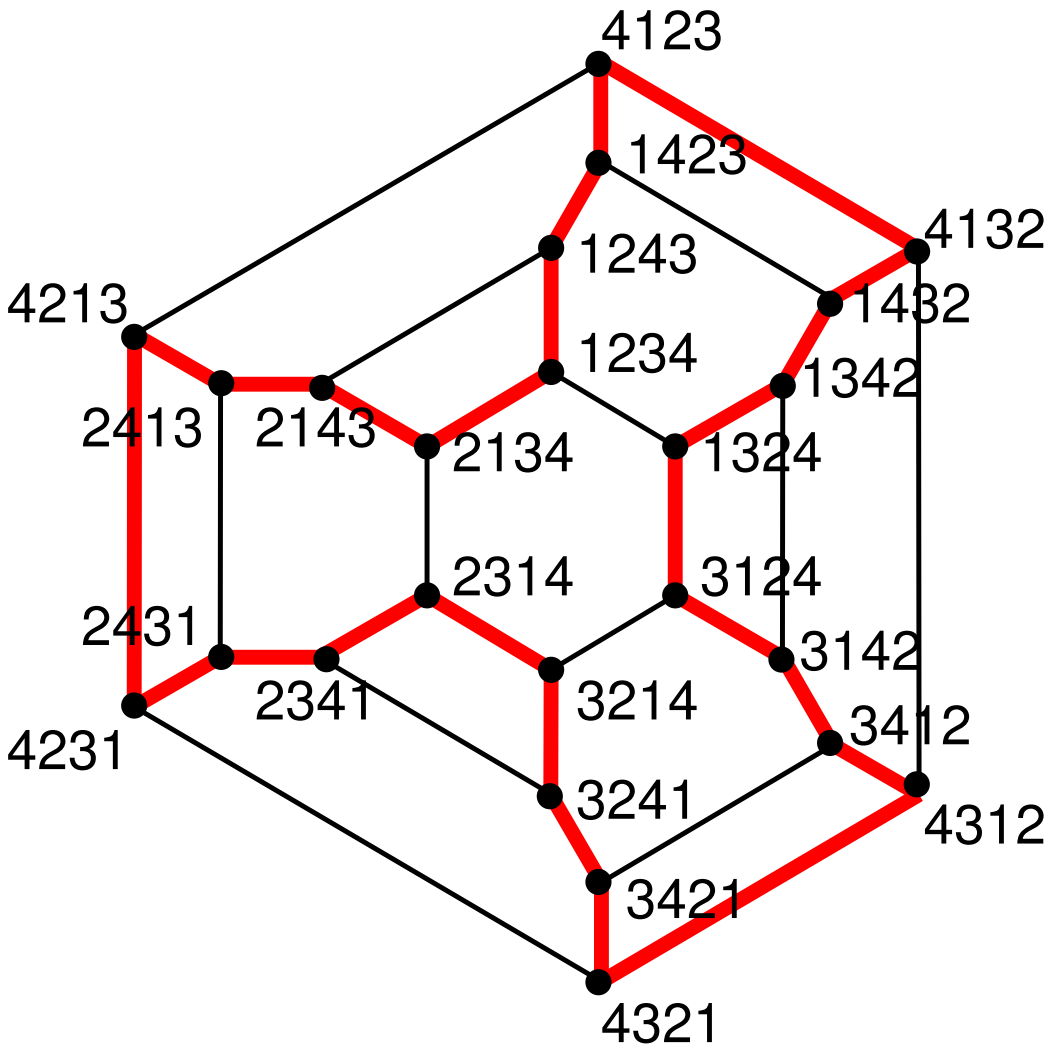
- 根付き木，親子関係の定義
- 親から子を見つける方法

順列列挙問題に対するグラフ P_n



a, a' **が** P_n **で隣接** $\Leftrightarrow$ **隣接要素の交換で** a **から** a' **が得られる**

順列列挙問題に対するグラフ P_n のハミルトン閉路



a, a' が P_n で隣接 $\Leftrightarrow$ 隣接要素の交換で a から a' が得られる

組合せ Gray 符号で次の順列を見つける方法

- $\{1, \ldots, n-1\}$ に対する組合せ的 Gray コードを用意する

123　312　231

132　321　213

これは Steinhaus–Johnson–Trotter アルゴリズムと呼ばれる

組合せ Gray 符号で次の順列を見つける方法

- $\{1, \ldots, n-1\}$ に対する組合せ的 Gray コードを用意する
- n を可能な場所に順に挿入する
- 右から左へ，左から右へ，右から左へ，... と順に

1234 312 231
1243
1423
4123
132 321 213

これは Steinhaus–Johnson–Trotter アルゴリズムと呼ばれる

組合せ Gray 符号で次の順列を見つける方法

- $\{1, \ldots, n-1\}$ に対する組合せ的 Gray コードを用意する
- n を可能な場所に順に挿入する
- 右から左へ，左から右へ，右から左へ，... と順に

1234	312	231
1243		
1423		
4123		
4132	321	213
1432		
1342		
1324		

これは Steinhaus–Johnson–Trotter アルゴリズムと呼ばれる

組合せ Gray 符号で次の順列を見つける方法

- $\{1, \ldots, n-1\}$ に対する組合せ的 Gray コードを用意する
- n を可能な場所に順に挿入する
- 右から左へ，左から右へ，右から左へ，... と順に

1234	3124	231
1243	3142	
1423	3412	
4123	4312	
4132	321	213
1432		
1342		
1324		

これは Steinhaus–Johnson–Trotter アルゴリズムと呼ばれる

組合せ Gray 符号で次の順列を見つける方法

- $\{1, \ldots, n-1\}$ に対する組合せ的 Gray コードを用意する
- n を可能な場所に順に挿入する
- 右から左へ，左から右へ，右から左へ，... と順に

1234	3124	231
1243	3142	
1423	3412	
4123	4312	
4132	4321	213
1432	3421	
1342	3241	
1324	3214	

これは Steinhaus–Johnson–Trotter アルゴリズムと呼ばれる

組合せ Gray 符号で次の順列を見つける方法

- $\{1, \ldots, n-1\}$ に対する組合せ的 Gray コードを用意する
- n を可能な場所に順に挿入する
- 右から左へ，左から右へ，右から左へ，... と順に

1234	3124	2314
1243	3142	2341
1423	3412	2431
4123	4312	4231
4132	4321	213
1432	3421	
1342	3241	
1324	3214	

これは Steinhaus–Johnson–Trotter アルゴリズムと呼ばれる

組合せ Gray 符号で次の順列を見つける方法

- $\{1, \ldots, n-1\}$ に対する組合せ的 Gray コードを用意する
- n を可能な場所に順に挿入する
- 右から左へ，左から右へ，右から左へ，... と順に

1234	3124	2314
1243	3142	2341
1423	3412	2431
4123	4312	4231
4132	4321	4213
1432	3421	2413
1342	3241	2143
1324	3214	2134

これは Steinhaus–Johnson–Trotter アルゴリズムと呼ばれる

組合せ Gray 符号で次の順列を見つける方法

- $\{1, \ldots, n-1\}$ に対する組合せ的 Gray コードを用意する
- n を可能な場所に順に挿入する
- 右から左へ，左から右へ，右から左へ，... と順に

1234	3124	2314
1243	3142	2341
1423	3412	2431
4123	4312	4231
4132	4321	4213
1432	3421	2413
1342	3241	2143
1324	3214	2134

- アルゴリズムでは n を動かせるだけ動かし，動かせなくなったら $n-1$ を 1 つ動かして，また n を動かし，... と続け，n も $n-1$ も動かせなくなったら $n-2$ を動かし，... と続けていく

これは Steinhaus–Johnson–Trotter アルゴリズムと呼ばれる

順列列挙問題へのアプローチ

分割探索によるアプローチ

- むしろ「バックトラック」と呼ぶものかもしれない

組合せ Gray 符号によるアプローチ

- グラフの定義，ハミルトン道の存在性
- ハミルトン道上の探索法

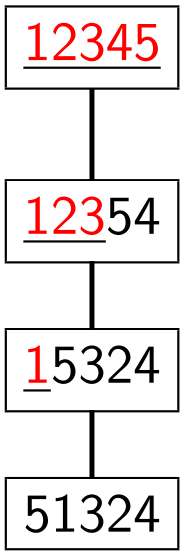
逆探索によるアプローチ

- 根付き木，親子関係の定義
- 親から子を見つける方法

逆探索の基本戦略 (順列列挙問題)

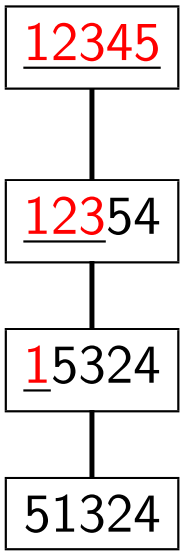
親子関係による根付き木の定義

- **根：** $(1, 2, \ldots, n)$
- **順列** a **の親：** $a_i \neq i$ **であるような最小の** i **に関して，**
  $a_j = i$ **として，** a_i **と** a_j **を交換した順列**



親子関係による根付き木の定義

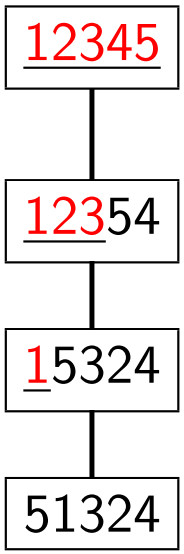
- **根：** $(1, 2, \ldots, n)$
- **順列 a の親：** $a_i \neq i$ **であるような最小の** i **に関して，** $a_j = i$ **として，** a_i **と** a_j **を交換した順列**
- **根付き木は** well-defined **：**
  根に近いほど $a_i = i$ **を満たす接頭辞** (prefix) **が長い**



逆探索の基本戦略 (順列列挙問題)

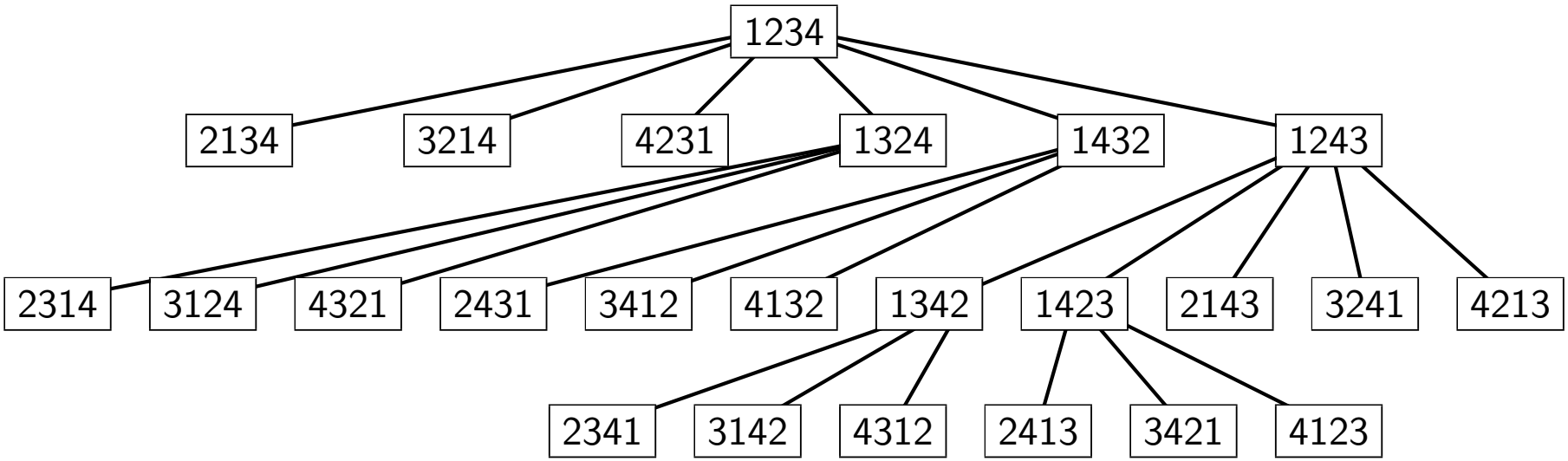
親子関係による根付き木の定義

- **根：** $(1, 2, \ldots, n)$
- **順列 a の親：** $a_i \neq i$ **であるような最小の** i **に関して，**
  $a_j = i$ **として，** a_i **と** a_j **を交換した順列**
- **根付き木は** well-defined**：**
  根に近いほど $a_i = i$ **を満たす接頭辞**
  (prefix) **が長い**
- **順列 a の子：**
  $a_i \neq i$ **であるような最小の** i **に関して，**
  $a_{i'}$ **と** $a_{j'}$ **を交換した順列**
  ただし， $i' < i$ **かつ** $i' < j'$
  ($i = 1$ **のとき，** a **は葉**)
  (i **が存在しないとき，** $i = n{+}1$ **とする**)



逆探索の基本戦略 (順列列挙問題)：列挙木の例

$n = 4$ の場合



逆探索による順列列挙アルゴリズム

順列列挙アルゴリズム (逆探索)

入力：自然数 n，出力：$\{1, \ldots, n\}$ の順列すべて

- $\mathrm{B}((1, 2, \ldots, n), n+1)$ **を呼び出し**

$\mathrm{B}(a, i)$

事前条件：a は $\{1, \ldots, n\}$ の順列，$i = \max\{j \mid a_k = k \ \forall k \leq j\} + 1$
事後条件：列挙木における a と a の子孫をすべて出力

- a **を出力**
- $i = 1$ **ならば終了**
- **そうでなければ，$i' := 1, j' := 2$ として以下を繰り返し**
  - $a' := a$ **から** $a_{i'}$ **と** $a_{j'}$ **を交換して得られる順列**
  - $\mathrm{B}(a', i')$ **を呼び出し**
  - $i' = i - 1$ **かつ** $j' = n$ **ならば終了**
  - $j' = n$ **ならば** $i' := i'+1, j' := i'+1$**，** $j' \neq n$ **ならば** $j' := j'+1$

例を見てきました

部分集合列挙問題 (済)

入力：自然数 n
出力：集合 $\{1, 2, \ldots, n\}$ の部分集合すべて

順列列挙問題 (済)

入力：自然数 n
出力：集合 $\{1, 2, \ldots, n\}$ の順列すべて

他の例

- **後続の講義と演習問題で**
- **皆さん自身の持っている問題で**

パート 1 の内容

- **列挙問題，列挙アルゴリズムとは?**
- **列挙問題，列挙アルゴリズム設計の難しさ**
- **列挙アルゴリズム設計技法**
  - **分割探索** (binary partition)
  - **組合せ** Gray **符号** (combinatorial Gray code)
  - **逆探索** (reverse search)
- **難しい列挙問題**

効率的な列挙アルゴリズムが作れない問題??

がんばってもできなかったら，それなりの理由があるはず

例

- 1 つ見つけることすら難しい問題
- 今までの出力に漏れがないか判定することすら難しい問題
- 今までの出力に重複があるか判定することすら難しい問題

部分集合和列挙問題

入力：n 個の自然数 $a_1, \ldots, a_n$，もう 1 個自然数 b
出力：$\sum_{i \in S} a_i = b$ となる $S \subseteq \{1, \ldots, n\}$ すべて

- **1 つでも出力する問題が NP 困難** (Karp 1972)
- $\xrightarrow{\text{なので}}$ **列挙も難しい**

注

- NP **困難問題はたくさん知られている**
- **列挙アルゴリズムを考える前に**
  「1 つ見つけるのがどれほど難しいか」考えるのが重要
  (それによりアルゴリズム設計指針が大きく変わる)

今までの出力に漏れがないかの判定が難しい問題

凸多面集合の頂点列挙問題

入力：自然数 d と d 次元空間中の n 個の半空間
出力：半空間の共通部分 (凸多面集合) の頂点すべて

- **この問題に出力多項式時間アルゴリズムがあれば** $\mathrm{P} = \mathrm{NP}$
  (Khachiyan, Boros, Borys, Elbassioni, Gurvich 2006)
- **別の言い方をすると：**
  この問題の入力と出力の一部が与えられたとき，
  それ以外に出力すべきものがあるか判定する問題が NP **困難**
- $\xrightarrow{\text{なので}}$ **列挙は難しい**

注

- **同様な結果が極大** t**-頻出集合問題に対しても知られている**
  (Boros, Gurvich, Khachiyan, Makino 2003)

見つかった 2 つが「重複」かどうかの判定が難しい問題

ラベル無しグラフ列挙問題

入力：自然数 n
出力：頂点数 n のラベル無しグラフすべて

- **新しく見つかったグラフが今までに見つかったグラフと同じ (同型) かどうか判定できれば十分**
- $\xrightarrow{\text{しかし}}$ **グラフ同型性判定問題は難しい**
  (多項式時間で解けるか，NP 完全か知られていない)
- $\xrightarrow{\text{なので}}$ **アルゴリズム構築が難しい**

注

- **同様な問題に「線形符号列挙問題」がある**
- **同型性判定が簡単なグラフクラスが多く知られているが，そのアルゴリズムは「標準型」を用いることが多い．標準型は列挙にも適用できる (中野先生の講義)**

パート 1 の内容：まとめ

- 列挙問題，列挙アルゴリズムとは?
  - 列挙アルゴリズムの効率：出力多項式時間，多項式時間遅延
- 列挙問題，列挙アルゴリズム設計の難しさ
- 列挙アルゴリズム設計技法
  - 分割探索 (binary partition)
  - 組合せ Gray 符号 (combinatorial Gray code)
  - 逆探索 (reverse search)
- 難しい列挙問題

C 言語，Python で岡本が書いた汚いプログラム集はこちらを参照

`www.jaist.ac.jp/~okamotoy/lect/2011/enumschool/`

プログラムはこの講義のアルゴリズムに基づくが，
この講義で想定したほど高速な実装にはなっていないので注意

演習問題

演習問題をやることはとても重要です

- 大学のアルゴリズムの授業で，アルゴリズムを自分で考える機会がほとんどないのは非常に残念
- 演習ではアルゴリズムを自分で考えて，アルゴリズム設計の楽しさを味わいましょう

演習問題への Tips

- 1 人でやらずにグループを作ってやりましょう
- グループでは解法のあらすじを議論しましょう
- 解法の詳細は 1 人で詰めてみましょう
- 不明な点はどんどん講師に聞きましょう
- 1 つの問題にこだわりすぎず，いろんな問題を試しましょう
- 演習問題を全部やる必要はありません

参考文献 (1/3)

列挙アルゴリズム設計技法については

- **宇野毅明,「列挙問題」，応用数理計画ハンドブック** (**久保，田村，松井編**)，**第** 14.4 **節，朝倉書店，**2002，pp. 886–932**.**

簡単な対象の列挙については

- A. Nijenhuis and H.S. Wilf, "Combinatorial Algorithms," Academic Press, 1978.
  (Herbert Wilf **の** Web **ページより全文ダウンロード可能**)
- D. Knuth, "The Art of Computer Programming," Vol. 4A, Addison-Wesley, Upper Saddle River, NJ, 2011.
  (**分冊** (fascicles) **の日本語訳あり**)
- **仙波一郎,「組合せアルゴリズム」，サイエンス社，**1989**.**

組合せ Gray **符号については**

- C. Savage, A survey of combinatorial Gray codes. SIAM Review **39** (1997) 605–629.

逆探索については

- D. Avis and K. Fukuda, Reverse search for enumeration. Discrete Applied Mathematics **65** (1996) 21–46.
- **福田公明,「逆探索とその応用」，離散構造とアルゴリズム II (藤重編)，共立出版，**1993，pp. 47–78.
- J.L. White, Reverse search for enumeration — applications. 2008. http://cgm.cs.mcgill.ca/~avis/doc/rs/applications/
- J.L. White, Reverse search for enumeration — implementations. http://cgm.cs.mcgill.ca/~avis/doc/rs/implementations/

前後順序探索については

- Knuth, TAOCP Vol. 4, Fac. 4.
- M. Sekanina, On an ordering of the set of vertices of a connected graph. Spisy Přírodovědecké Fakulty University v Brně **412** (1960) 137–140.

その他，スライド中で引用した文献

- R.M. Karp, Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher (eds), Complexity of Computer Computations, New York, Plenum, pp. 85–103.
- L. Khachiyan, E. Boros, K. Borys, K. Elbassioni and V. Gurvich, Generating all vertices of a polyhedron is hard. Discrete & Computational Geometry **39** (2006) 174–190.
- E. Boros, V. Gurvich, L. Khachiyan and K. Makino, On maximal frequent and minimal infrequent sets in binary matrices. Annals of Mathematics and Artificial Intelligence **39** (2003) 211–221.