

I482F 実践的アルゴリズム特論
(11) 厳密アルゴリズム：分枝限定法

岡本 吉央
okamotoy@jaist.ac.jp

北陸先端科学技術大学院大学

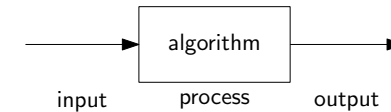
2011年6月25日

”最終更新：2011/06/24 14:59”

アルゴリズムとは

アルゴリズムがすること (簡単に言うと)

- ▶ データを受け取り (入力)
- ▶ それを処理し (処理)
- ▶ 結果を返す (出力)



アルゴリズムに要求されること

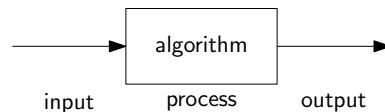
- ▶ どんな入力に対しても (入力の普遍性)
- ▶ 高速に処理し (処理の高速性)
- ▶ 正しい結果を返す (出力の正当性)

この3つの要求を緩和できる場合がある

緩和の仕方 (2) : 「処理の高速性」の緩和

アルゴリズムに要求されること

- ▶ どんな入力に対しても (入力の普遍性)
- ▶ **高速に処理し** (処理の高速性)
- ▶ 正しい結果を返す (出力の正当性)



「処理の高速性」の緩和

- ▶ どんな入力に対しても (入力の普遍性)
- ▶ **ある程度の速さで処理し** (処理の高速性の緩和)
- ▶ 正しい結果を返す (出力の正当性)

分枝限定法とは？

分枝限定法 (branch-and-bound method)

系統的な場合分けによって解を探索する方法
大きく、2つの部分から構成される

- ▶ 分枝操作 (branching) : 1つの入力を複数の入力に分割する
- ▶ 限定操作 (bounding) : 分枝操作が不必要である場合に打ち切る

この講義の目標

次の理解

- ▶ 分枝限定法の典型例を理解
- ▶ 分枝限定法の性能解析を理解

① 厳密アルゴリズムと分枝限定法

② 頂点被覆問題

分枝限定法に基づくアルゴリズム 1
分枝限定法に基づくアルゴリズム 2

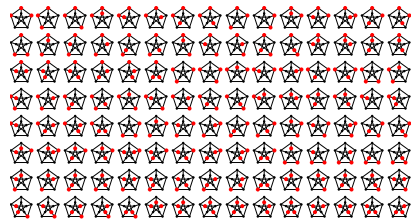
③ 計算木の葉数の解析

④ 今日のまとめと補足

すぐに思いつくアルゴリズム

- ① 要素数 k の頂点部分集合 $S \subseteq V$ すべてに対して S が G の頂点被覆であるか判定
- ② 1 つでも頂点被覆となる S が存在すれば, Yes を出力
- ③ そうでなければ, No を出力

例: $n = 10, k = 3$ の場合



▶ 反復回数 = $\binom{n}{k} = O(n^k)$
 $(n = 10, k = 3 \text{ のとき}, \binom{10}{3} = 120)$

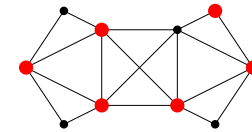
例として用いる問題: 頂点被覆問題

頂点被覆問題 (判定問題)

- ▶ 入力: 無向グラフ $G = (V, E)$, 自然数 k
- ▶ 出力: G が要素数 k 以下の頂点被覆を持てば「Yes」, そうでなければ「No」

G の頂点被覆 (vertex cover):

頂点部分集合 S で, どの辺の端点も 1 つは S に含まれるもの



頂点被覆である

① 厳密アルゴリズムと分枝限定法

② 頂点被覆問題

分枝限定法に基づくアルゴリズム 1
分枝限定法に基づくアルゴリズム 2

③ 計算木の葉数の解析

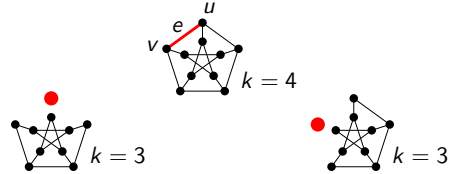
④ 今日のまとめと補足

分枝限定法に基づくアルゴリズム 1：基本アイデア

- ▶ 任意の辺 e に着目
- ▶ e の端点の 1 つは必ず頂点被覆に含まれる，したがって

観察 1

G に要素数 k の頂点被覆が存在 \Leftrightarrow
 G から e の端点を除去したグラフに要素数 $k - 1$ の頂点被覆が存在



問題点

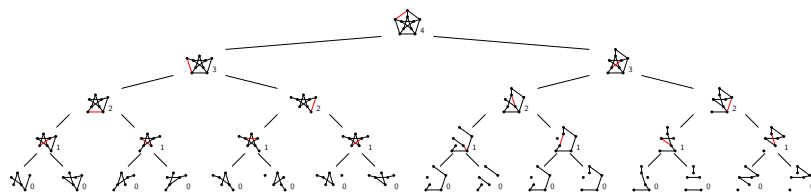
e のどちらの端点を除去したグラフに存在するのか分からない
 (よって、両方とも試す)

分枝限定法に基づくアルゴリズム 1

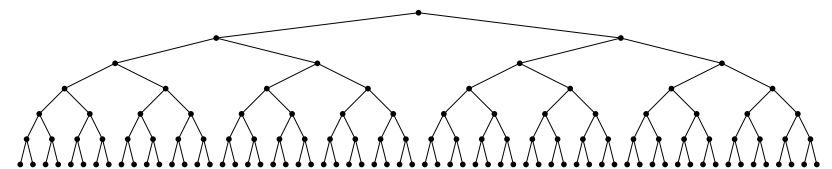
アルゴリズム BB1(G, k)

- ① G に辺がないならば，Yes を出力して停止
- ② G に辺があり， $k \leq 0$ ならば，No を出力して停止
- ③ // G に辺があり， $k \geq 1$ のとき
 - ① $e = \{u, v\} \leftarrow G$ の任意の辺
 - ② BB1($G - u, k - 1$) か BB1($G - v, k - 1$) が Yes を出力するならば，Yes を出力して停止
 - ③ そうでなければ，No を出力して停止

BB1：実行例

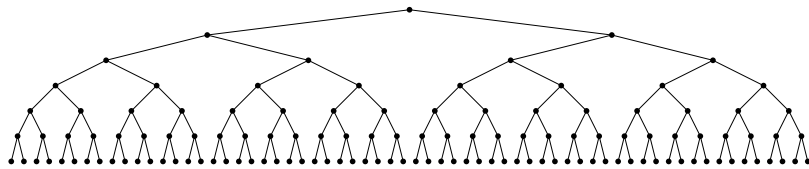


BB1：計算木



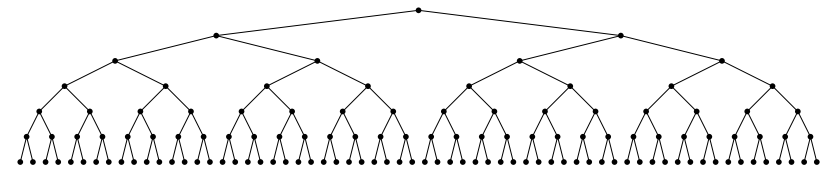
- ▶ 1 つのノードがアルゴリズムの実行を表す
- ▶ ノードの子はそのアルゴリズムの再帰的実行を表す

BB1: 計算量と計算木の関係



- ▶ 調べ上げる場合の数 = 計算木の葉の数
- ▶ ∴ 計算木の葉の数が計算量に直接影響を与える
- ▶ (∴ 計算木の葉の数がいくつなのか知ることが重要)

BB1: 計算木の葉の数



- ▶ 入力を G, k としたときの計算木の葉の数は次の式を満たす $L(k)$ 以下

$$L(k) = \begin{cases} 1 & (k = 0 \text{ のとき}) \\ 2L(k-1) & (k > 0 \text{ のとき}) \end{cases}$$

- ▶ この再帰式を解く: $k > 0$ のとき

$$L(k) = 2L(k-1) = 2 \cdot 2L(k-2) = \dots = 2^k \cdot L(0) = 2^k$$

- ▶ ∴ 計算木の葉の数は 2^k 以下

単純なアルゴリズムと分枝限定アルゴリズムの比較

- ▶ 単純なアルゴリズムが調べ上げる場合の数 = $\binom{n}{k}$
- ▶ 分枝限定アルゴリズムが調べ上げる場合の数 = 2^k

$n = 1000$ の場合

k	$\binom{n}{k}$	2^k
2	499,500	4
3	166,167,000	8
4	41,417,124,750	16
5	8,250,291,250,200	32
6	1,368,173,298,991,500	64
7	194,280,608,456,793,000	128
8	24,115,080,524,699,431,125	256
9	2,658,017,764,500,203,964,000	512
10	263,409,560,461,970,212,832,400	1,024
11	23,706,860,441,577,319,154,916,000	2,048

① 厳密アルゴリズムと分枝限定法

② 頂点被覆問題

- 分枝限定法に基づくアルゴリズム 1
- 分枝限定法に基づくアルゴリズム 2

③ 計算木の葉数の解析

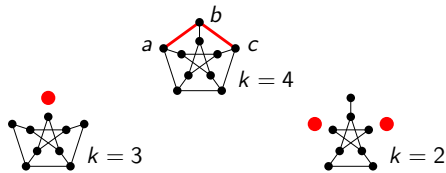
④ 今日のまとめと補足

分枝限定法に基づくアルゴリズム 2 : 基本アイデア

- ▶ 長さ 2 のパス $a-b-c$ に着目
- ▶ 頂点被覆には b が含まれるか、あるいは $\{a, c\}$ が含まれるしたがって、

観察 2

G に要素数 k の頂点被覆が存在 \Leftrightarrow
 $G - b$ に要素数 $k - 1$ の頂点被覆が存在、または、
 $G - \{a, c\}$ に要素数 $k - 2$ の頂点被覆が存在



分枝限定法に基づくアルゴリズム 2

アルゴリズム BB2(G, k)

- ① G に長さ 2 のパスがないならば、次のスライドの記述に従う
- ② G に長さ 2 のパスがあり、 $k \leq 0$ ならば、No を出力して停止
- ③ // G に長さ 2 のパスがあり、 $k \geq 1$ のとき
 - ① $a-b-c \leftarrow G$ の任意の長さ 2 のパス
 - ② BB1($G - b, k - 1$) か BB1($G - \{a, c\}, k - 2$) が Yes を出力するならば、Yes を出力して停止
 - ③ そうでなければ、No を出力して停止

分枝限定法に基づくアルゴリズム 2 (続き)

アルゴリズム BB2(G, k)

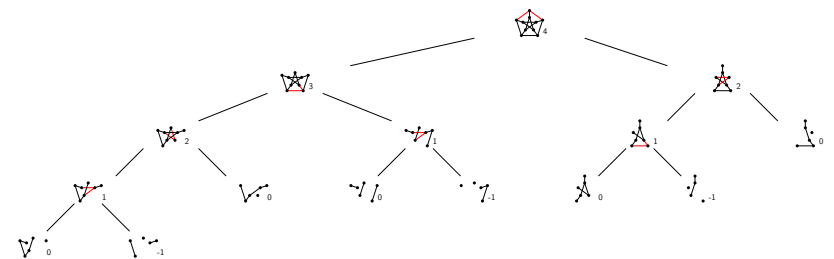
- ① G に長さ 2 のパスがないならば、
 - ① G の辺の数 $\leq k$ ならば、Yes を出力して停止
 - ② そうでなければ、No を出力して停止



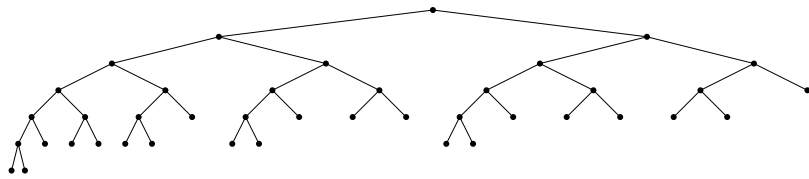
観察 3

G に長さ 2 のパスがない $\Rightarrow G$ の各頂点の次数 ≤ 1

BB2 : 実行例

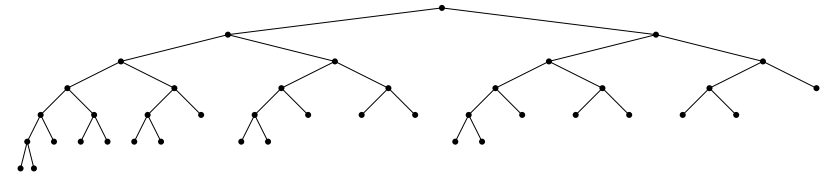


BB2 : 計算木



- ▶ 右の部分木に対応する再帰呼び出しでは k が $k - 2$ になるため BB1 に比べて木が小さくなっている
- ▶ すなわち、葉の数も小さくなっている

BB2 : 計算木の葉の数



- ▶ 入力を G, k としたときの計算木の葉の数は次の式を満たす $L(k)$ 以下

$$L(k) = \begin{cases} 1 & (k \leq 0 \text{ のとき}) \\ L(k-1) + L(k-2) & (k > 0 \text{ のとき}) \end{cases}$$

- ▶ この再帰式を解く (後述) : $k > 0$ のとき

$$L(k) = O(1.619^k)$$

- ▶ ∴ 計算木の葉の数は $O(1.619^k)$ 以下

単純なアルゴリズムと分枝限定アルゴリズムの比較

- ▶ 分枝限定アルゴリズム 1 が調べ上げる場合の数 = 2^k
- ▶ 分枝限定アルゴリズム 2 が調べ上げる場合の数 = $O(1.619^k)$

$n = 1000$ の場合

k	2^k	$\lceil 1.619^k \rceil$
2	4	3
3	8	5
4	16	7
5	32	12
6	64	19
7	128	30
8	256	48
9	512	77
10	1,024	124
11	2,048	201

① 厳密アルゴリズムと分枝限定法

② 頂点被覆問題

分枝限定法に基づくアルゴリズム 1
分枝限定法に基づくアルゴリズム 2

③ 計算木の葉数の解析

④ 今日のまとめと補足

再帰式をどう解くか？

解きたい再帰式

$$L(k) = \begin{cases} 1 & (k \leq 0 \text{ のとき}) \\ L(k-1) + L(k-2) & (k > 0 \text{ のとき}) \end{cases}$$

考え方

- ▶ $L(k) = \alpha^k$ と置いて, α を定める
- ▶ その α に対して, $L(k) = O(\alpha^k)$

再帰式をどう解くか？ (2)

- ▶ $L(k) = \alpha^k$ と置くと, $k > 0$ のとき

$$\alpha^k = \alpha^{k-1} + \alpha^{k-2}$$

- ▶ つまり,

$$\alpha^2 = \alpha + 1$$

- ▶ これを α に関する方程式だと思って解くと

$$\alpha_+ = \frac{1 + \sqrt{5}}{2}, \quad \alpha_- = \frac{1 - \sqrt{5}}{2}$$

という2つの解が得られる (α_+ は正, α_- は負)

再帰式をどう解くか？ (3)

主張

このとき, 任意の $k \geq 0$ に対して, $L(k) \leq 2\alpha_+^k$ となる

証明: 帰納法による

- ▶ $k = 0$ のとき: $L(0) = 1 \leq 2\alpha_+^0$ (done)
- ▶ $k = 1$ のとき: $L(1) = 2 \leq 2\alpha_+^1$ (done)
- ▶ $k \geq 2$ のとき: 任意の $k' < k$ に対して $L(k') \leq 2\alpha_+^{k'}$ と仮定
- ▶ このとき

$$L(k) = L(k-1) + L(k-2) \leq 2\alpha_+^{k-1} + 2\alpha_+^{k-2} = 2(\alpha_+^{k-1} + \alpha_+^{k-2}) = 2\alpha_+^k$$

□

よって, $L(k) \leq 2 \left(\frac{1+\sqrt{5}}{2} \right)^k = O(1.619^k)$

他の再帰

仮に, 頂点被覆問題の分枝限定アルゴリズムの解析で, 次の再帰式を解く必要が出てきたとする

再帰式 2

$$L(k) = \begin{cases} 1 & (k \leq 0 \text{ のとき}) \\ L(k-2) + L(k-3) + L(k-5) & (k > 0 \text{ のとき}) \end{cases}$$

同じように解こうとすると...

他の再帰：どう解くか (1)

- ▶ $L(k) = \alpha^k$ と置くと, $k > 0$ のとき

$$\alpha^k = \alpha^{k-2} + \alpha^{k-3} + \alpha^{k-5}$$

が得られ, よって, α に関する次の方程式が得られる

$$\alpha^5 = \alpha^3 + \alpha^2 + 1$$

- ▶ これを (数値的に) 解くと, 正の実数解は

$$\alpha_+ = 1.430$$

の唯一つ (ただし, 小数点以下第 4 位で切り上げ)

他の再帰：どう解くか (2)

主張

このとき, 任意の $k \geq 0$ に対して, $L(k) \leq 7\alpha_+^k$ となる

証明：帰納法による

- ▶ $k = 0$ のとき: $L(0) = 1 \leq 7\alpha_+^0$ (done)
- ▶ $k = 1$ のとき: $L(1) = 3 \leq 7\alpha_+^1$ (done)
- ▶ ...
- ▶ $k = 4$ のとき: $L(4) = 7 \leq 7\alpha_+^4$ (done)
- ▶ $k \geq 5$ のとき: 任意の $k' < k$ に対して $L(k') \leq 7\alpha_+^{k'}$ と仮定
- ▶ このとき

$$\begin{aligned} L(k) &= L(k-2) + L(k-3) + L(k-5) \leq 7\alpha_+^{k-2} + 7\alpha_+^{k-3} + 7\alpha_+^{k-5} \\ &= 7(\alpha_+^{k-2} + \alpha_+^{k-3} + \alpha_+^{k-5}) = 7\alpha_+^k \end{aligned}$$

□

よって, $L(k) \leq 7 \cdot 1.430^k = O(1.430^k)$

葉数の解析が示唆する分枝限定アルゴリズムの設計法

たとえば

次の再帰式を満たす $L_1(k), L_2(k), L_3(k)$ の中で, どれが一番小さいか

- ① $L_1(k) = 2L_1(k-1)$
- ② $L_2(k) = 3L_2(k-1)$
- ③ $L_3(k) = 2L_3(k-2)$

- ▶ $L_1(k)$ と $L_2(k)$ を比べると, $L_1(k)$ の方が小さい
(右辺の係数「2」と「3」の比較)
... 再帰呼び出しの回数の比較
- ▶ $L_1(k)$ と $L_3(k)$ を比べると, $L_3(k)$ の方が小さい
(右辺のカッコの中「-1」と「-2」の比較)
... 再帰呼び出しにおける k の変化の比較

- ① 厳密アルゴリズムと分枝限定法
- ② 頂点被覆問題
分枝限定法に基づくアルゴリズム 1
分枝限定法に基づくアルゴリズム 2
- ③ 計算木の葉数の解析
- ④ 今日のまとめと補足

今日のまとめ

分枝限定法

系統的な場合分けによって解を探索する方法

大きく、2つの部分から構成される

- ▶ 分枝操作 (branching) : 1つの入力を複数の入力に分割する
- ▶ 限定操作 (bounding) : 分枝操作が不必要である場合に打ち切る

頂点被覆問題を例として

- ▶ すぐに思いつくアルゴリズム : $O(n^k)$
- ▶ 辺に着目した分枝限定法 : $O(2^k)$
- ▶ 長さ2のパスに着目した分枝限定法 : $O(1.619^k)$

計算木による計算量の解析

- ▶ 葉の数の上界を表す再帰式 $L(k) = L(k-1) + L(k-2)$
- ▶ $L(k) = \alpha^k$ として、方程式 $\alpha^k = \alpha^{k-1} + \alpha^{k-2}$ を解く

参考文献

- ▶ 厳密アルゴリズムの設計と理論的解析
 - ▶ F.V. Fomin, D. Kratsch. Exact Exponential Algorithms. Springer, 2010.
 - ▶ R. Niedermeier. Invitation to Fixed-Parameter Algorithms. Oxford University Press, 2006.
 - ▶ 岡本吉央. 離散最適化に対する高速な厳密アルゴリズム. オペレーションズ・リサーチ 50 (2005) 763-769.
 - ▶ 岡本吉央. 固定パラメータ容易性. オペレーションズ・リサーチ 52 (2007) 535-537.
- ▶ 分枝限定アルゴリズムの理論的解析
 - ▶ O. Kullmann. Fundamentals of branching heuristics. In: Handbook of Satisfiability, Chapter 7, 2009, pp. 205-244.