# 離散最適化基礎論 第 9 回
# 計算量 (2)：大規模近傍

**岡本 吉央**
okamotoy@uec.ac.jp

**電気通信大学**

2013 年 12 月 20 日

**最終更新：**2013 年 12 月 29 日　18:34

# 目次

# 局所探索法を評価する観点

## 局所探索法 (local search)：一般的枠組み

0. 許容解 $x \in S$ を適当に見つける (この $x$ を初期解と呼ぶ)
1. 以下を繰り返し
   1. ある $x' \in N(x)$ が存在して $f(x') < f(x)$ ならば $x \leftarrow x'$
   2. そうでなければ繰り返しを終了
2. $x$ を出力 (この $x$ は $N$ に関する局所最適解 (定理 1.1))

局所探索法の出力は $N$ に関する局所最適解

## 局所探索法を評価する観点

- **性能保証** (performance guarantee)
  局所最適解が最適解に (目的関数値で) どれだけ近いか？
- **計算量** (computational complexity)
  局所探索法がどれだけ速く局所最適解に収束するか？

# 局所探索法を評価する観点：計算量の評価

## 局所探索法 (local search)：一般的枠組み

0. 許容解 $x \in S$ を適当に見つける (この $x$ を初期解と呼ぶ) ←時間 $T_A$
1. 以下を繰り返し ←反復回数 $k$
   1. ある $x' \in N(x)$ が存在して $f(x') < f(x)$ ならば ←時間 $T_C$
      $x \leftarrow x'$
   2. そうでなければ繰り返しを終了
2. $x$ を出力 (この $x$ は $N$ に関する局所最適解 (定理 1.1))

## 全体の計算時間

およそ $T_A + k \cdot T_C$

## よって，局所探索法が多項式時間アルゴリズムになるには，次が必要

- $T_A$ が多項式
- $k \cdot T_C$ が多項式

# 局所探索法を評価する観点：計算量の評価

## 局所探索法 (local search)：一般的枠組み

0. 許容解 $x \in S$ を適当に見つける (この $x$ を初期解と呼ぶ) ←時間 $T_A$
1. 以下を繰り返し ←反復回数 $k$
   1. ある $x' \in N(x)$ が存在して $f(x') < f(x)$ ならば ←時間 $T_C$
      $x \leftarrow x'$
   2. そうでなければ繰り返しを終了
2. $x$ を出力 (この $x$ は $N$ に関する局所最適解 (定理 1.1))

## 全体の計算時間

およそ $T_A + k \cdot T_C$

## よって，局所探索法が最悪の場合指数時間かかるには，次が十分

- $T_A$ が指数関数
- $k$ が指数関数
- $T_C$ が指数関数

# 局所探索法の計算量評価

## 局所探索法 (local search)：一般的枠組み

0. 許容解 $x \in S$ を適当に見つける (この $x$ を初期解と呼ぶ) ←時間 $T_A$
1. 以下を繰り返し ←反復回数 $k$
   1. ある $x' \in N(x)$ が存在して $f(x') < f(x)$ ならば ←時間 $T_C$
      $x \leftarrow x'$
   2. そうでなければ繰り返しを終了
2. $x$ を出力 (この $x$ は $N$ に関する局所最適解 (定理 1.1))

## ポイント 1

反復回数の評価

## ポイント 2

反復継続条件の確認にかかる時間の評価
(近傍探索時間の評価)

# スケジュール

**近傍探索時間の評価**



**反復回数の評価**



**近似局所探索による反復回数の削減**



**局所探索法にこだわらない局所最適化の難しさ**

# 前回の内容

## いままでの感覚

**許容解 $x$ の近傍 $N(x)$ の中で，$x$ よりもよい許容解を見つけるためには，$|N(x)|$ に比例する時間は必要**

## 前回と今回のテーマ

**実はそうではなく，$|N(x)|$ よりも小さな時間で見つけられる (または，存在しないことが分かる)**

**前回の例**

- **最終完了時刻最小化スケジューリングの移動近傍**
  $O(mn) \rightsquigarrow O(\log n)$
- **巡回セールスマン問題の** 2opt **近傍**
  $O(n^2) \rightsquigarrow O(n \log n)$

**鍵となる概念**

- **データ構造**

# 今回の内容

## いままでの感覚

**許容解 $x$ の近傍 $N(x)$ の中で，$x$ よりもよい許容解を見つけるためには，$|N(x)|$ に比例する時間は必要**

## 前回と今回のテーマ

**実はそうではなく，**
**$|N(x)|$ よりも小さな時間で見つけられる (または，存在しないことが分かる)**

**今回の例 (超大規模近傍)**

- **巡回セールスマン問題の割当近傍** (Sarvanov, Doroshko '81)
  $O((n/2)!) \rightsquigarrow O(n^3)$
- **巡回セールスマン問題のピラミッド型巡回路近傍** (Carlier, Villon '90)
  $O(2^n) \rightsquigarrow O(n^2)$

**鍵となる概念**

- **効率よく解ける離散最適化問題**

# 超大規模近傍

## 超大規模近傍とは？

離散最適化問題の近傍関数 $N\colon S \to 2^S$ が**超大規模** であるとは，
任意の $c \in \mathbb{N}$ に対して，次が成り立つこと

$$\max_{x \in S} |N(x)| > (\text{入力データの大きさ})^c$$

つまり，近傍の大きさが入力データの大きさの多項式よりも大きい
(例えば，指数関数的に大きい)

- 前回扱った近傍関数は超大規模ではない
- 今回扱う近傍関数は超大規模である

## 注意

超大規模近傍関数に対して，ナイーブな近傍探索を行うと，
時間がかかりすぎる

ナイーブでない方法を使えば，効率よく近傍探索が行える場合がある

# 目次

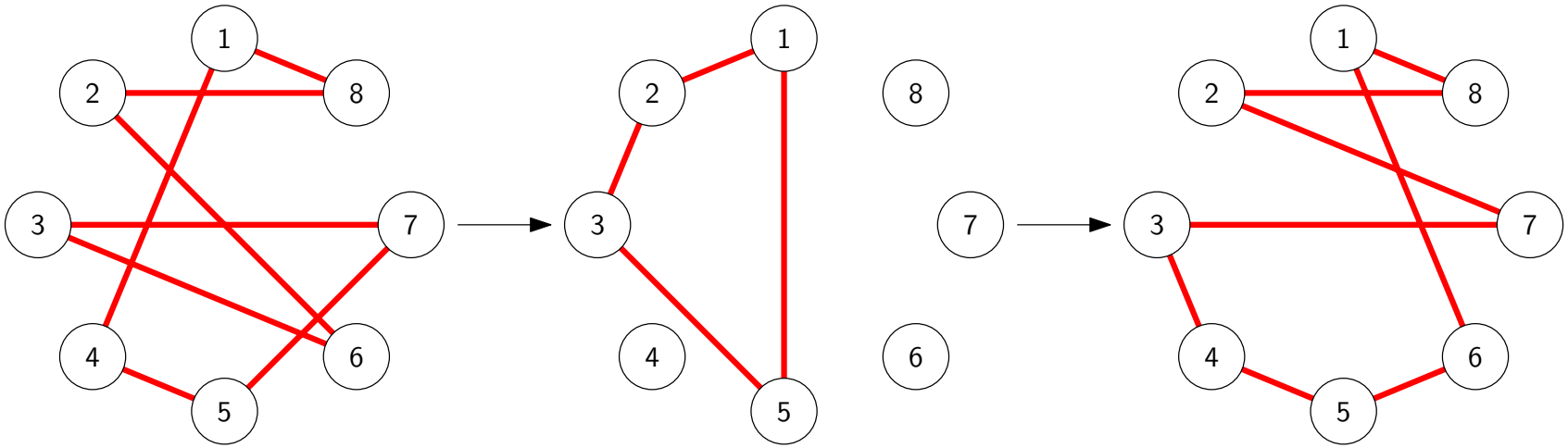
# 巡回セールスマン問題の割当近傍

**都市数 $n$：偶数**

## 巡回セールスマン問題に対する割当操作

$\tau = (\tau_1, \tau_2, \ldots, \tau_n)$：現在保持している巡回路

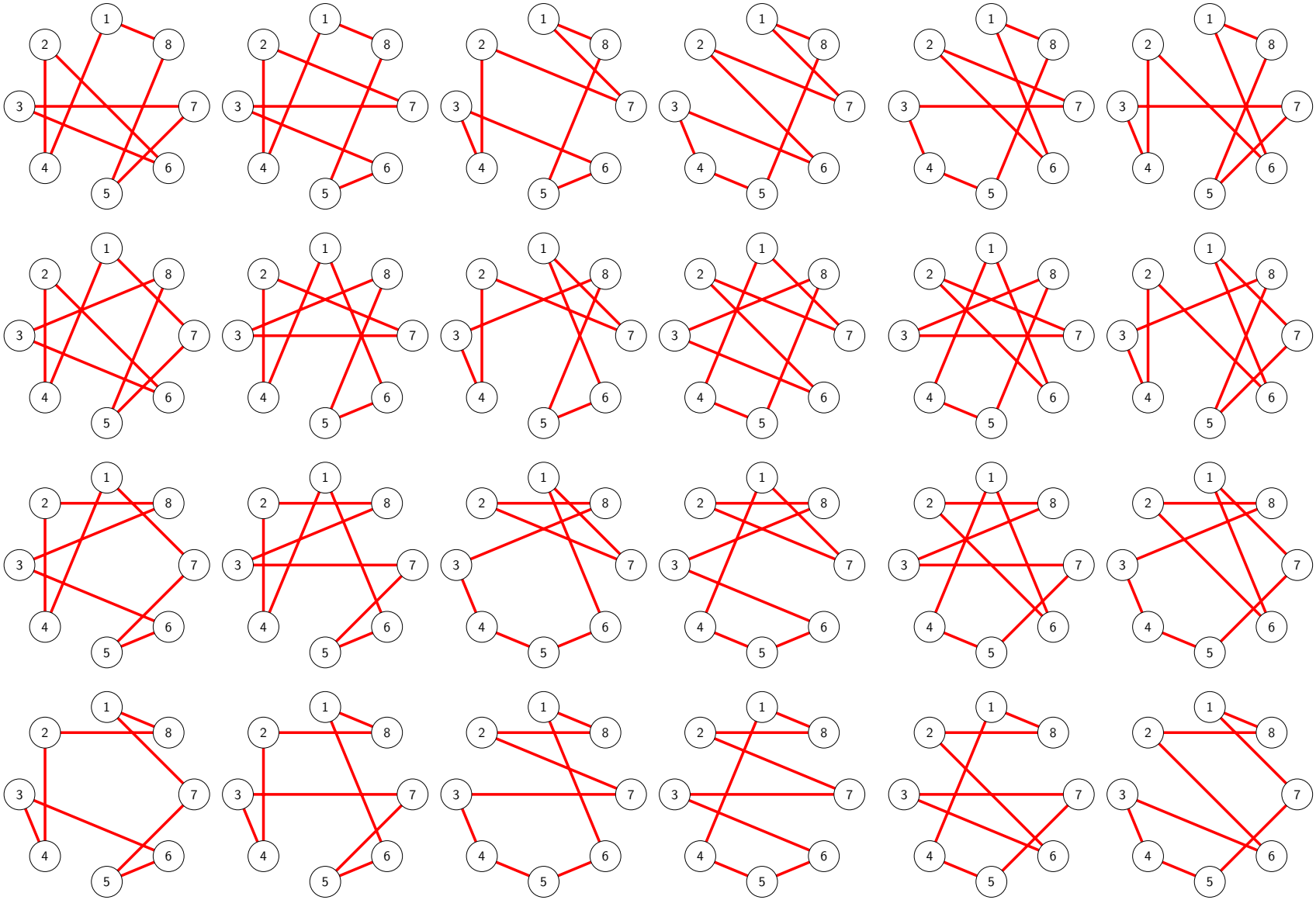
1. $\tau_2, \tau_4, \ldots, \tau_n$ を取り除く
2. 取り除いた各位置に取り除いた都市を 1 つ挿入する




$\tau = (1, 4, 5, 7, 3, 6, 2, 8)$　　$(1, \quad 5, \quad 3, \quad 2 \quad )$　　$\tau' = (1, 6, 5, 4, 3, 7, 2, 8)$

**割当操作が誘導する近傍を割当近傍と呼ぶ**

# 割当近傍の大きさ：例



# 割当近傍の大きさ

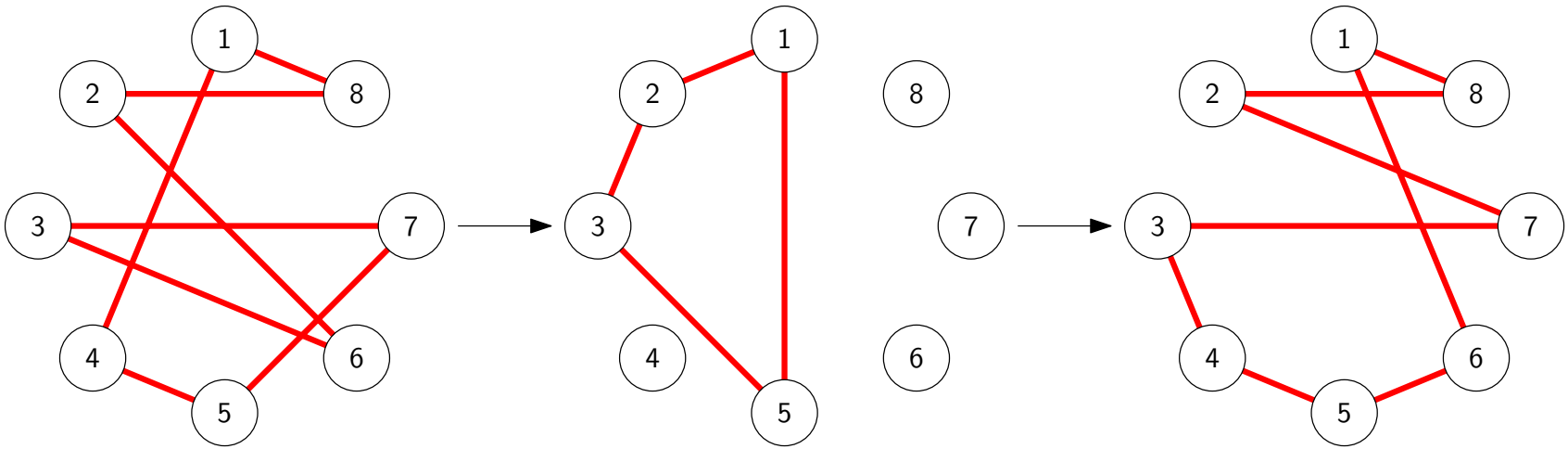
**都市数 $n$：偶数**

## 定理 9.1 (割当近傍の大きさ)

**任意の巡回路 $\tau$ の割当近傍 $N(\tau)$ の大きさは $(n/2)!$**

**証明**

- **除去した都市の数 $= n/2$**
- **それらを 1 列に並べる方法の数 $= (n/2)!$**
- **そのように並べる方法 $\leftarrow 1$ 対 $1 \rightarrow$ 除去した都市を挿入する方法** □



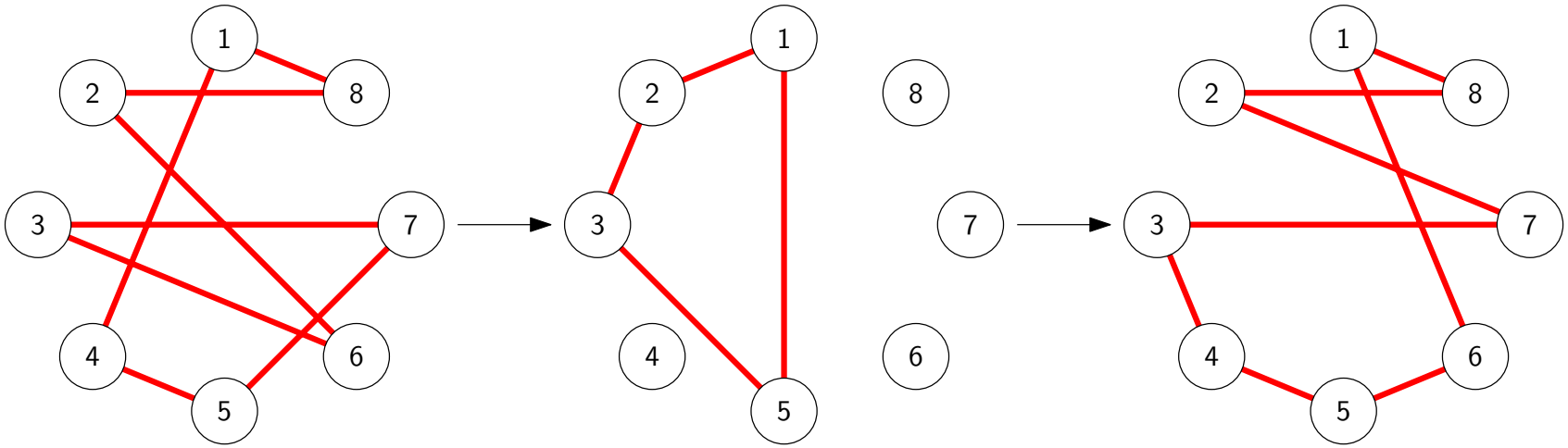

$\tau = (1, 4, 5, 7, 3, 6, 2, 8)$　　$(1, \quad 5, \quad 3, \quad 2 \quad )$　　$\tau' = (1, 6, 5, 4, 3, 7, 2, 8)$

# 割当近傍の探索

**都市数 $n$：偶数**

## 定理 9.2 (割当近傍の効率的探索)

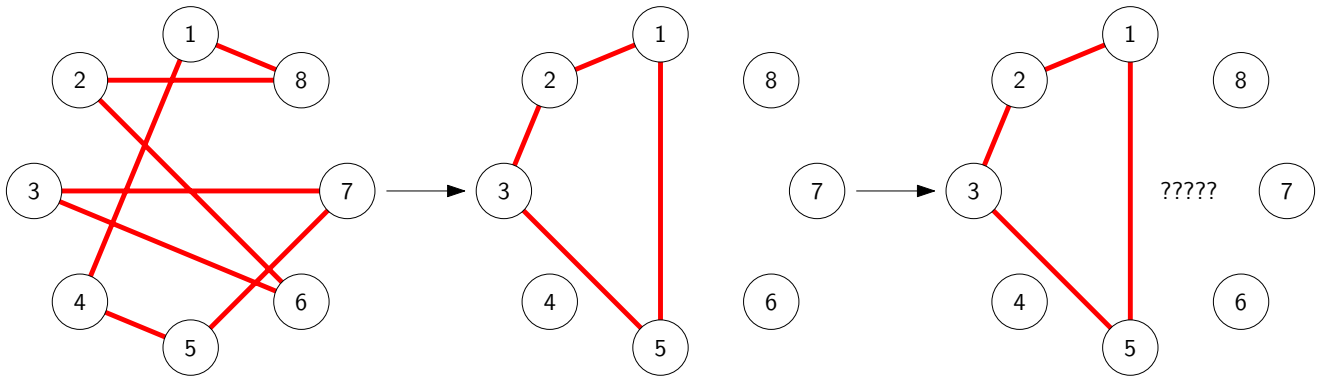
**任意の巡回路 $\tau$ の割当近傍 $N(\tau)$ において，**
**$\tau$ よりも短い巡回路があるか判定する問題は $O(n^3)$ 時間で解ける**
**また，そのような巡回路がある場合，同じ時間でそれを発見できる**

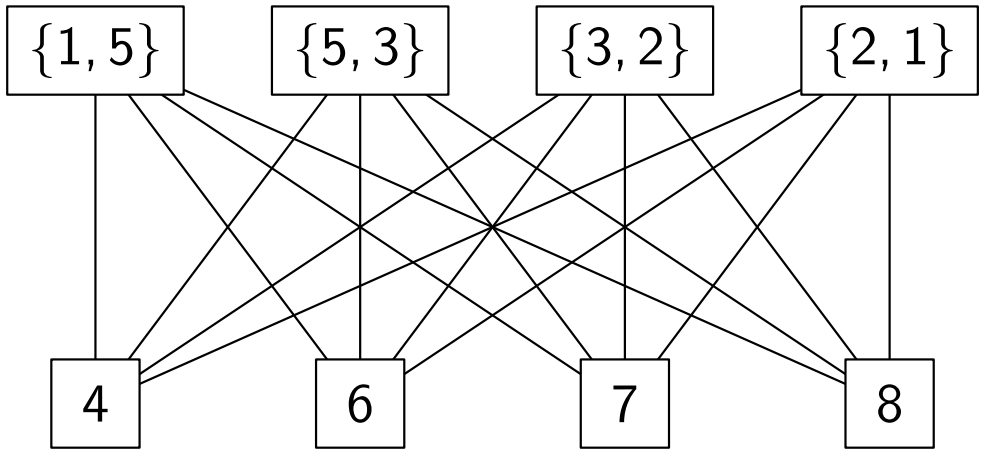



$\tau = (1, 4, 5, 7, 3, 6, 2, 8)$　　$(1, \quad 5, \quad 3, \quad 2 \quad )$　　$\tau' = (1, 6, 5, 4, 3, 7, 2, 8)$

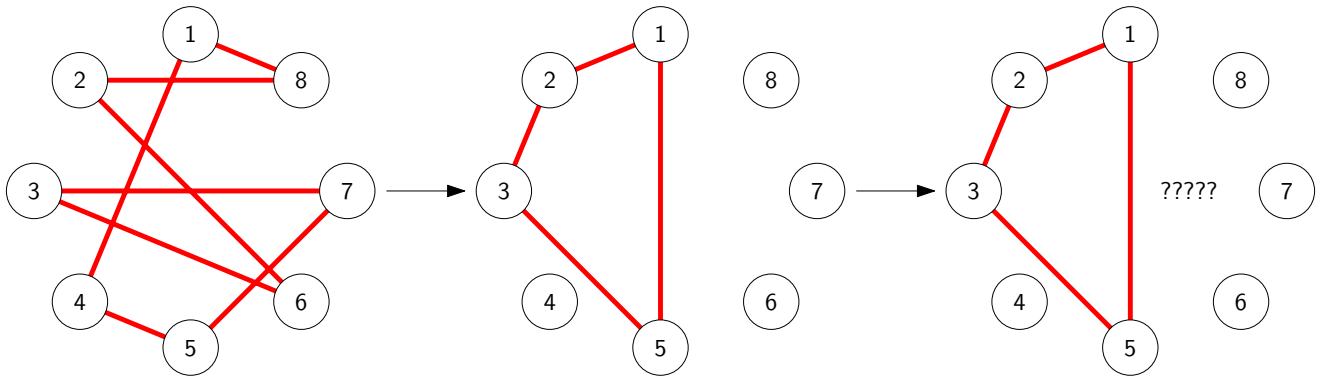
**証明の着想：「割当問題」に対するアルゴリズムを利用する**

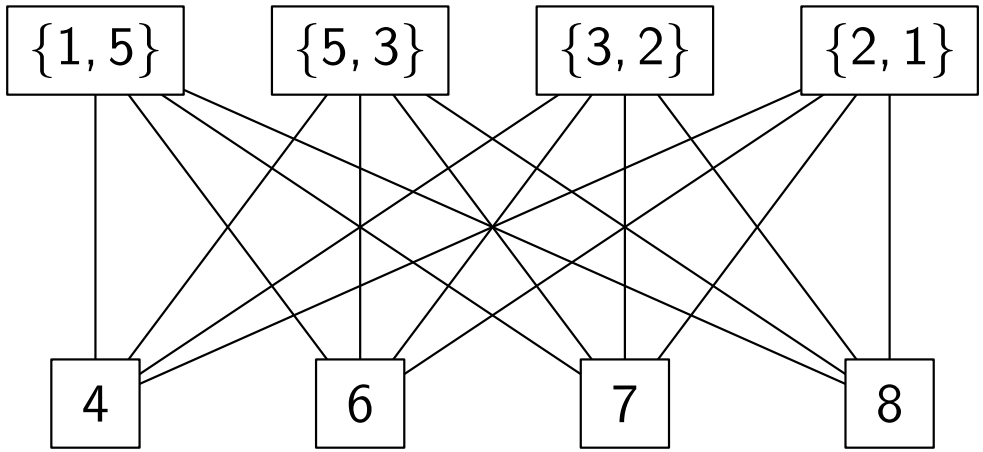
# 割当近傍の探索：辺重み付き二部グラフの構成 (1)



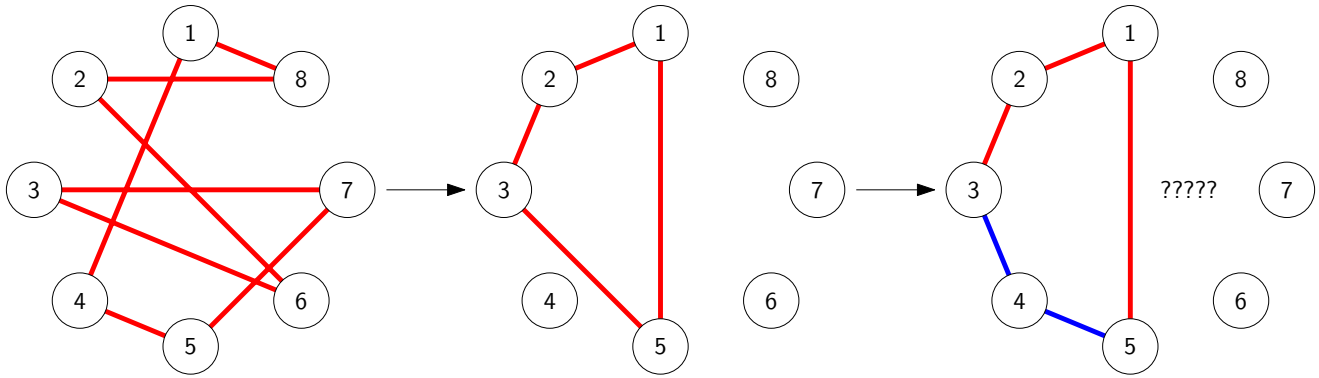
# 割当近傍の探索：辺重み付き二部グラフの構成 (1)

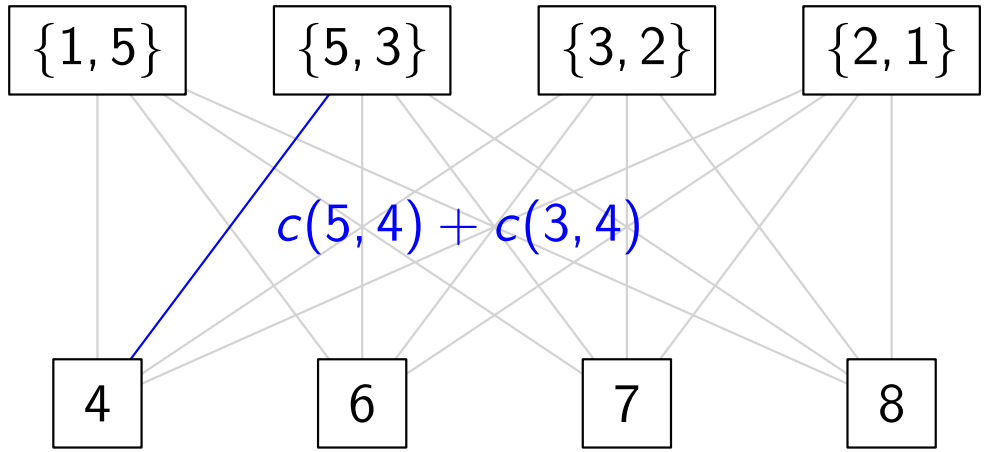



- 上には，都市除去後に残された巡回路の辺を並べる
- 下には，除去した都市を並べる

# 割当近傍の探索：辺重み付き二部グラフの構成 (2)

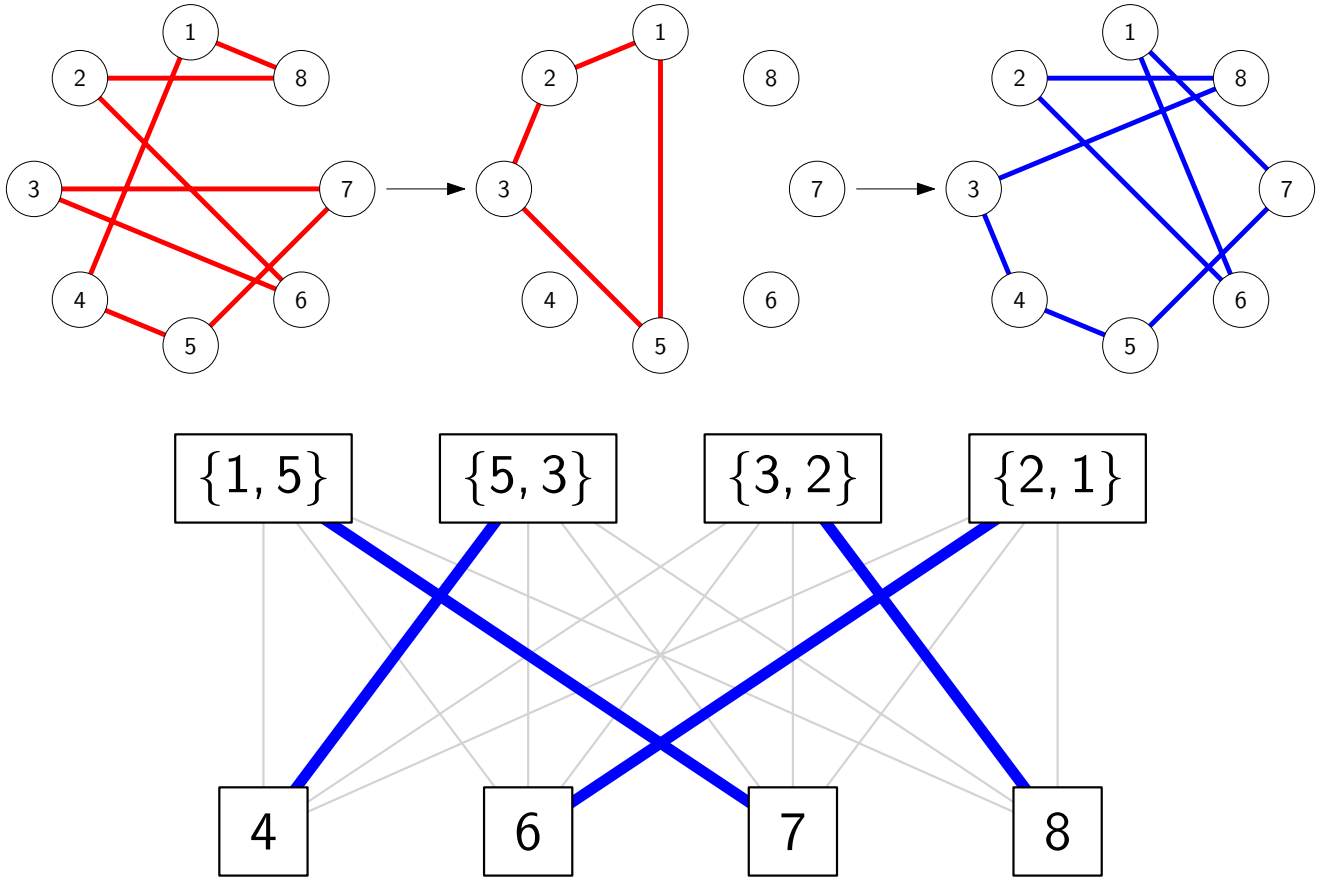



**上側の頂点 $\{i, j\}$ と下側の頂点 $k$ を結ぶ辺の重み $= c(i, k) + c(j, k)$**

- **これは，$i$ と $j$ の間に $k$ を挿入したときに生まれる長さ**

# 割当近傍の探索：辺重み付き二部グラフの構成 (3)




**上側の頂点と下側の頂点の間の 1 対 1 対応 $\leftrightarrow$ 割当操作**

**選ばれた辺の重みの総和 $\leftrightarrow$ 得られた巡回路の長さ**

# 行いたいこと

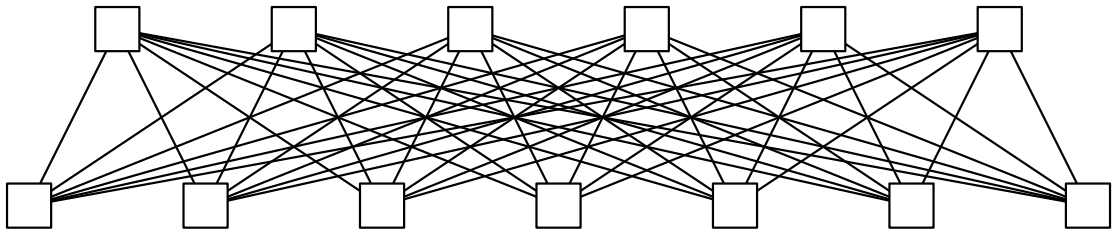
## 次の最適化問題を解きたい (割当問題と呼ばれる)

**入力**

- 辺重み付き (完全) 二部グラフ
  上側の頂点の数 $n_1$，下側の頂点の数 $n_2$ $(n_1 \leq n_2)$
- その二部グラフの各辺に対する重み (非負)

**出力**

- 上側の頂点を下側の頂点 1 つに対応付ける方法
  (上側から下側への単射) で，
  選ばれた辺の重み和が最小のもの



# 行いたいこと

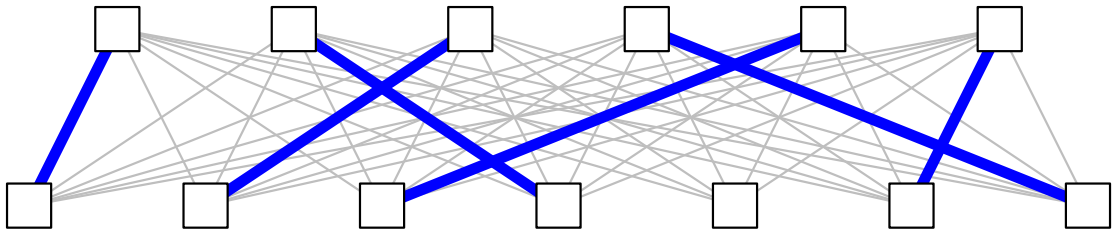
## 次の最適化問題を解きたい (割当問題と呼ばれる)

**入力**

- 辺重み付き (完全) 二部グラフ
  上側の頂点の数 $n_1$，下側の頂点の数 $n_2$ $(n_1 \leq n_2)$
- その二部グラフの各辺に対する重み (非負)

**出力**

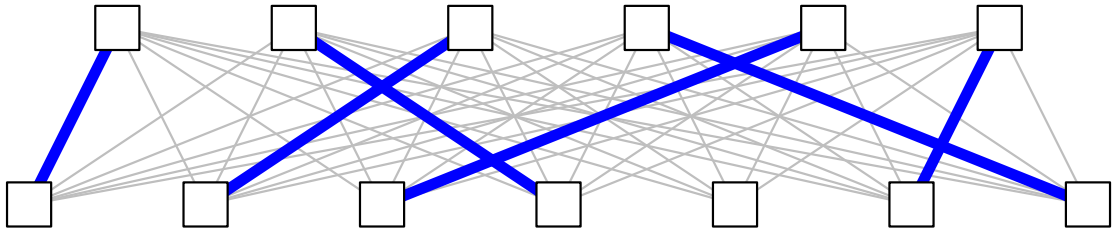
- 上側の頂点を下側の頂点 1 つに対応付ける方法
  (上側から下側への単射) で，
  選ばれた辺の重み和が最小のもの



# 割当問題に関する事実

## 事実

**割当問題は $O((n_1 + n_2)^3)$ 時間で解ける**



- ハンガリー法と呼ばれるアルゴリズムが知られている
- 最小費用流問題，線形計画問題としても解くことができる

# 割当近傍の探索：定理 9.2 の証明

$\tau = (\tau_1, \tau_2, \ldots, \tau_n)$：現在保持している巡回路，$n$ は偶数

## 割当近傍の効率的探索法

1. $\tau$ から $\tau_2, \tau_4, \ldots, \tau_n$ を除去する
2. 次のような辺重み付き二部グラフを作る
   - 上側には，都市除去後に残された巡回路の辺を並べる
   - 下側には，除去した都市を並べる
   - 上側の頂点 $\{i, j\}$ と下側の頂点 $k$ を結ぶ辺の重み $= c(i, k) + c(j, k)$
3. このグラフを入力とする割当問題を解く
4. その解を割当操作であると解釈して，都市を挿入する

# 割当近傍の探索：定理 9.2 の証明

$\tau = (\tau_1, \tau_2, \ldots, \tau_n)$：現在保持している巡回路，$n$ は偶数

## 割当近傍の効率的探索法

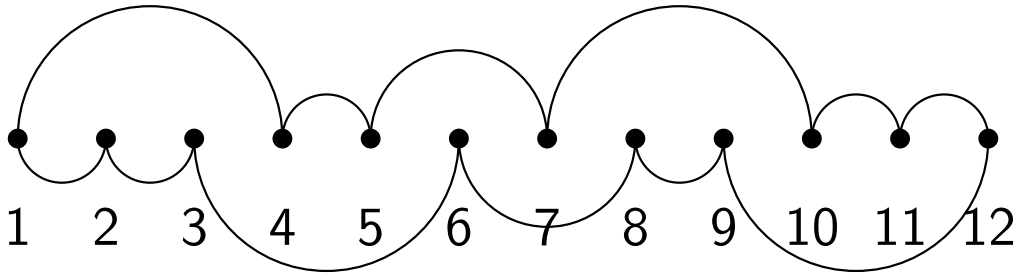
1. $\tau$ から $\tau_2, \tau_4, \ldots, \tau_n$ を除去する ← $O(n)$ 時間
2. 次のような辺重み付き二部グラフを作る ← $O(n^2)$ 時間
   - 上側には，都市除去後に残された巡回路の辺を並べる
   - 下側には，除去した都市を並べる
   - 上側の頂点 $\{i, j\}$ と下側の頂点 $k$ を結ぶ辺の重み $= c(i, k) + c(j, k)$
3. このグラフを入力とする割当問題を解く ← $O(n^3)$ 時間
4. その解を割当操作であると解釈して，都市を挿入する ← $O(n)$ 時間

全体の計算量：$O(n^3)$ 時間 □

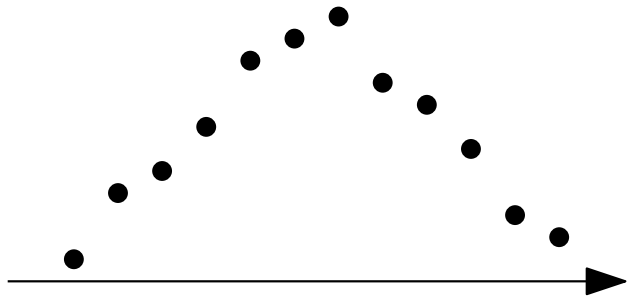
# 目次

# ピラミッド型巡回路：直感

**形が特殊な巡回路：** $\tau = (1, 4, 5, 7, 10, 11, 12, 9, 8, 6, 3, 2)$




**ピラミッドの気分を味わう**



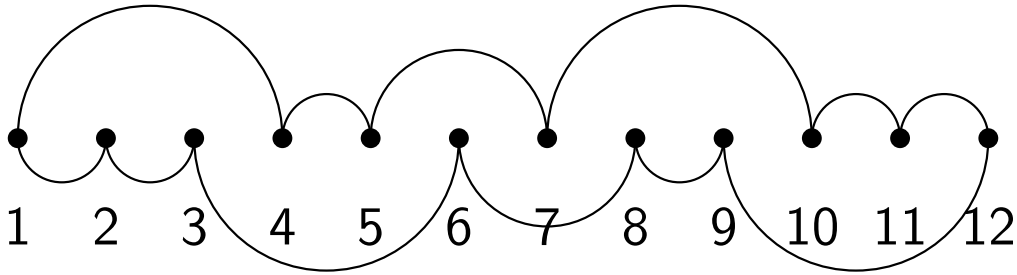
# ピラミッド型巡回路：定義

**都市の集合** $= \{1, 2, 3, \ldots, n\}$ **(**$n$ **は奇数でもよい)**

## ピラミッド型巡回路とは？

**巡回路** $\tau = (\tau_1, \tau_2, \ldots, \tau_n)$ **が**ピラミッド型**であるとは**
**ある** $j \in \{1, \ldots, n\}$ **が存在して，次が成り立つこと**

- $1 = \tau_1 < \tau_2 < \cdots < \tau_j = n$
- $\tau_j > \tau_{j+1} > \cdots > \tau_n$



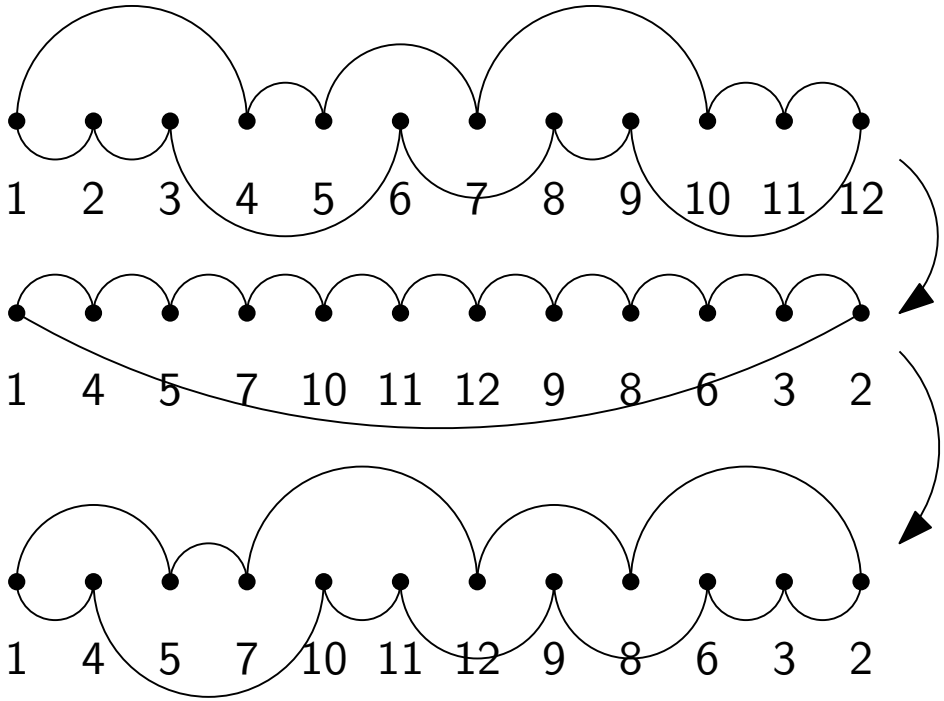

$\tau = (1, 4, 5, 7, 10, 11, 12, 9, 8, 6, 3, 2)$

# ピラミッド型巡回路を用いた局所操作

## ピラミッド型巡回路を用いた局所操作

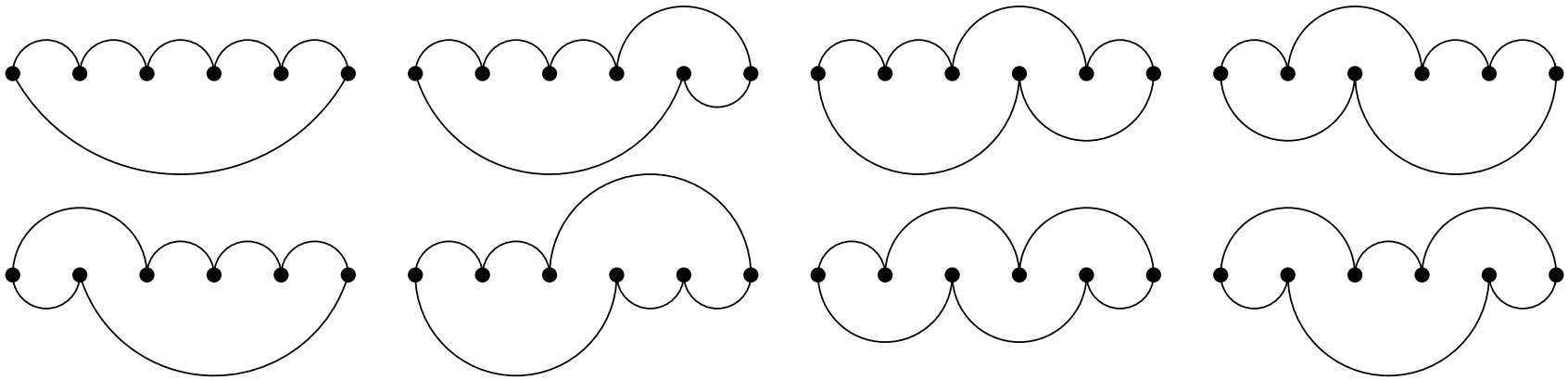
$\tau = (\tau_1, \tau_2, \ldots, \tau_n)$：現在保持している巡回路

1. $\tau_1 = 1, \tau_2 = 2, \ldots, \tau_n = n$ となるように添字を取り替える
2. ピラミッド巡回路を得る




この操作が誘導する近傍をピラミッド型巡回路近傍と呼ぶ

# ピラミッド型巡回路近傍の大きさ：例



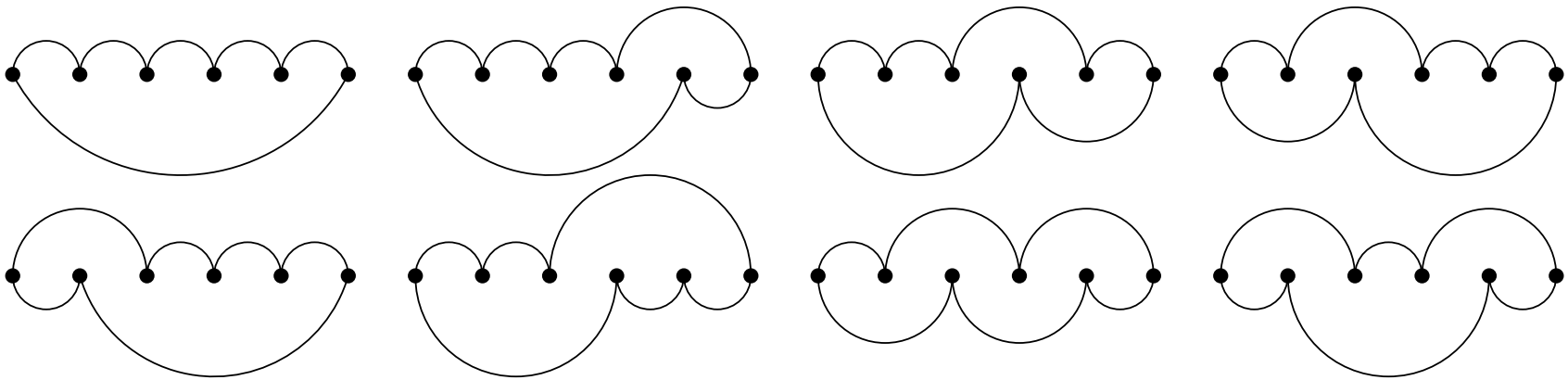
# ピラミッド型巡回路近傍の大きさ

$n$：都市数

**定理 9.3 (ピラミッド型巡回路近傍の大きさ)**

**任意の巡回路に対して，**
**そのピラミッド型巡回路近傍の大きさは $2^{n-3}$**

$n = 6$ **の例：** $2^{n-3} = 2^{6-3} = 8$



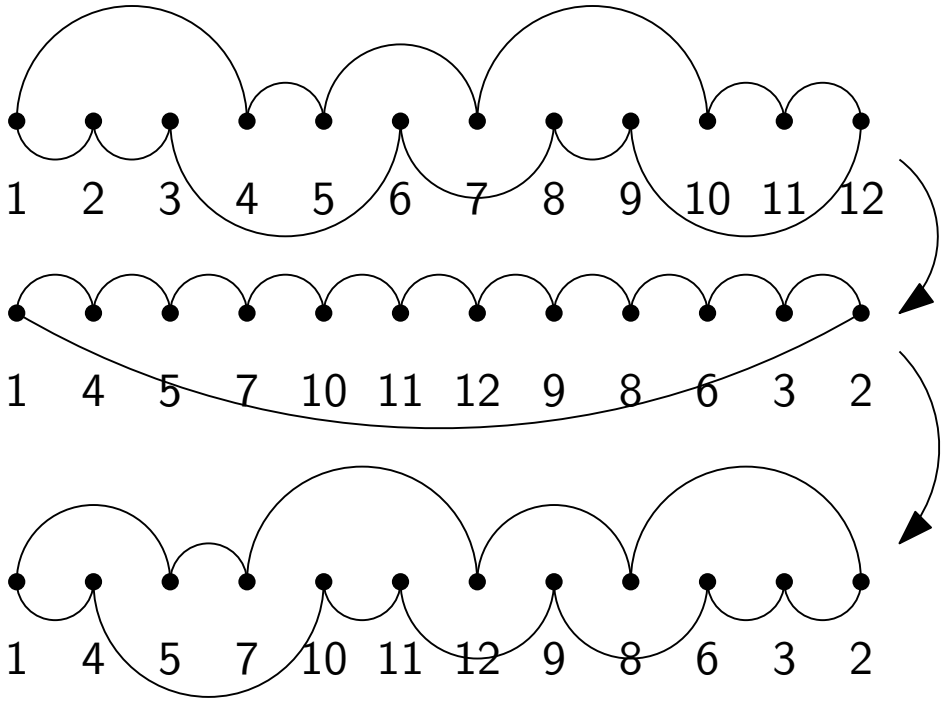
**証明：演習問題 (同じ巡回路を二度数えないように注意)**

# ピラミッド型巡回路近傍の探索

$n$：都市数

**定理 9.4 (ピラミッド型巡回路近傍の探索)**

**最短のピラミッド型巡回路は $O(n^2)$ 時間で発見できる**




**証明の着想：「動的計画法」を用いる**

# 動的計画法とは？

## 岩波数学辞典第 4 版 (2007) による説明 (の序論部)

最適化問題の中には，その過程がいくつもの段階から構成される多段決定問題と見なせるものが数多く存在する．動的計画法は，多段決定問題を体系的に取り扱う研究分野であり，1950 年以降に R. Bellman が発展させた理論・手法である．動的計画法は，離散最適化問題や組合せ問題に対しても，アルゴリズム設計のパラダイムとしてしばしば使われる．

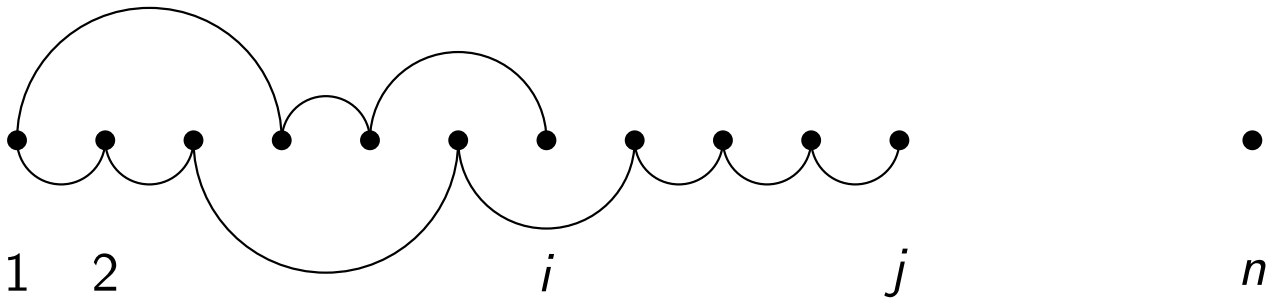
離散最適化問題に対するアルゴリズム設計パラダイムとして使うときは以下のような段階を踏む

- 解くべき問題を逐次的な問題に埋め込む
- 逐次的な問題の間の再帰関係を見出す
- その関係を用いて，解く

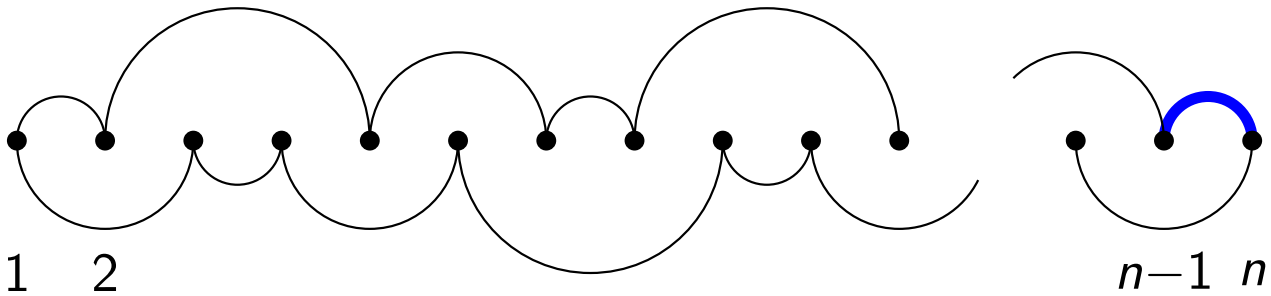
# ピラミッド型巡回路近傍の探索：逐次的な問題への埋め込み

**都市 $i$ と $j$ に対して (ただし，$2 \le i < j \le n$)**

- $D(i,j) =$ **都市 $\{1,\ldots,j\}$ をちょうど一度ずつ訪問する経路の中で，次を満たすものの最短長**
  - **$i$ から始まり，$j$ で終わる**
  - **$i$ から 1 まで，都市の番号が小さくなるように進む**
  - **1 から $j$ まで，都市の番号が大きくなるように進む**




**ピラミッド型巡回路の最短長** $= D(n-1,n) + c(n-1,n)$



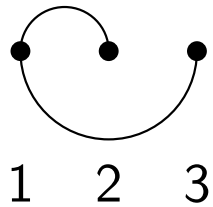
# ピラミッド型巡回路近傍の探索：再帰式 (1)

## 再帰式

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

**証明**：$i = 2,\ j = 3$ **のとき**

- **可能な経路は以下のものしかない**




- **その長さは** $c(1,2) + c(1,3)$

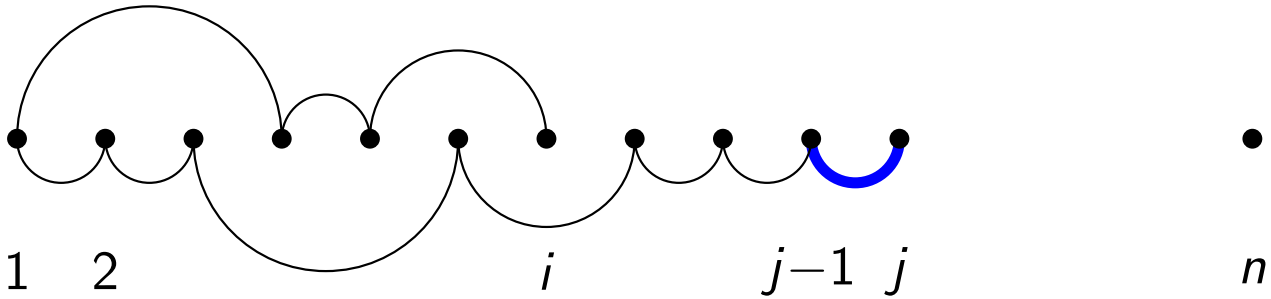
# ピラミッド型巡回路近傍の探索：再帰式 (2)

## 再帰式

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

**証明**：$i+1 < j$ のとき

- 可能な経路において，$j$ と $j-1$ は必ず辺で結ばれる




- そのような経路は $i$ から $j-1$ まで進み，
  その後で，$j-1$ から $j$ へ進む
- よって，最短長は $D(i,j-1) + c(j-1,j)$

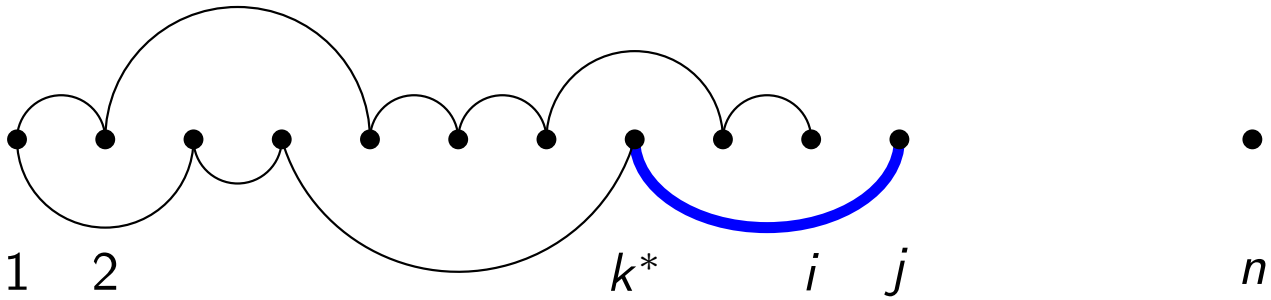
# ピラミッド型巡回路近傍の探索：再帰式 (3)

## 再帰式

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

**証明**：$i+1=j$ のとき

- $D(i,j)$ を与える経路にて，$j$ と $k^*$ が辺で結ばれるとする




- そのような経路は $k^*$ から $i$ まで進む経路を逆にたどり，その後で，$k^*$ から $j$ へ進む
- その長さは $D(k^*,i) + c(k^*,j)$

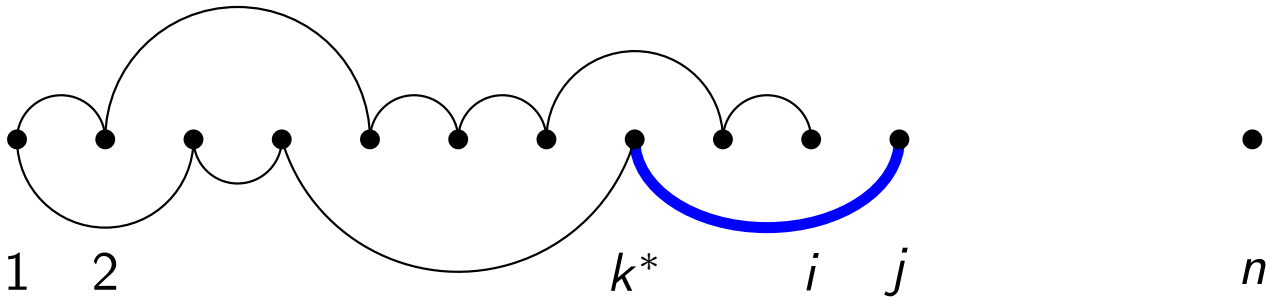
# ピラミッド型巡回路近傍の探索：再帰式 (4)

## 再帰式

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

**証明**：$i+1=j$ **のとき (続き)**

- $D(i,j)$ **を与える経路にて，** $j$ **と** $k^*$ **が辺で結ばれるとする**

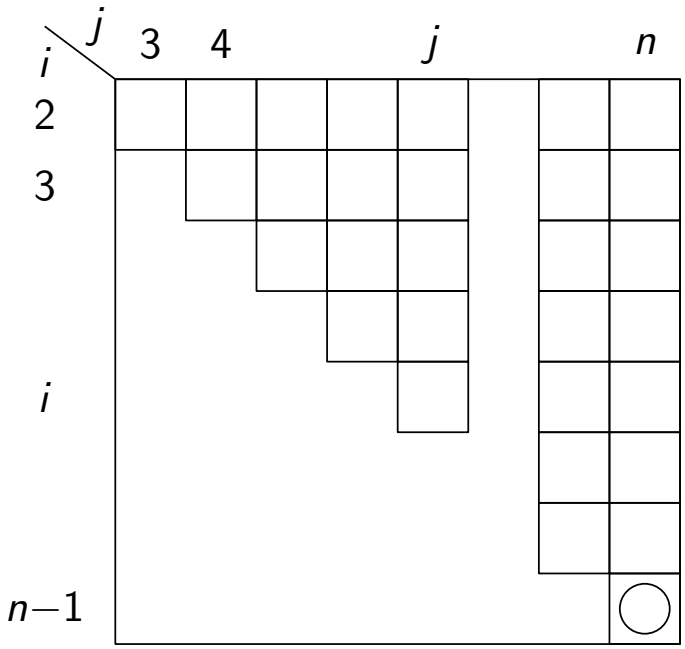



- $D(i,j)$ **の最小性から，他の都市** $k$ **(ただし，** $k<i$**) に対しては，** $D(k,i)+c(k,j) \geq D(k^*,i)+c(k^*,j)$ **が成り立つ**
- $\therefore\ D(i,j) = \min_{k<i}\{D(k,i) + c(k,j)\}$ □

# ピラミッド型巡回路近傍の探索：再帰式を解く (1)

## 再帰式 ($2 \leq i < j \leq n$)

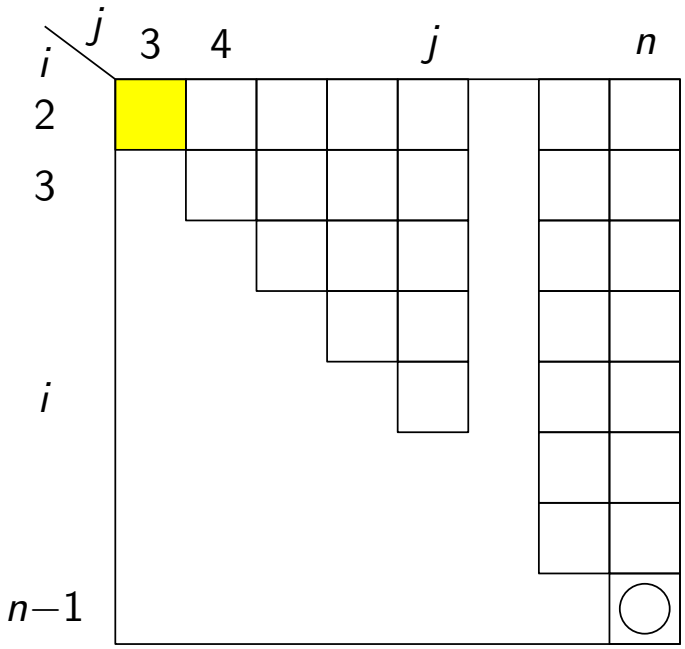
$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$



# ピラミッド型巡回路近傍の探索：再帰式を解く (1)

## 再帰式 ($2 \leq i < j \leq n$)

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

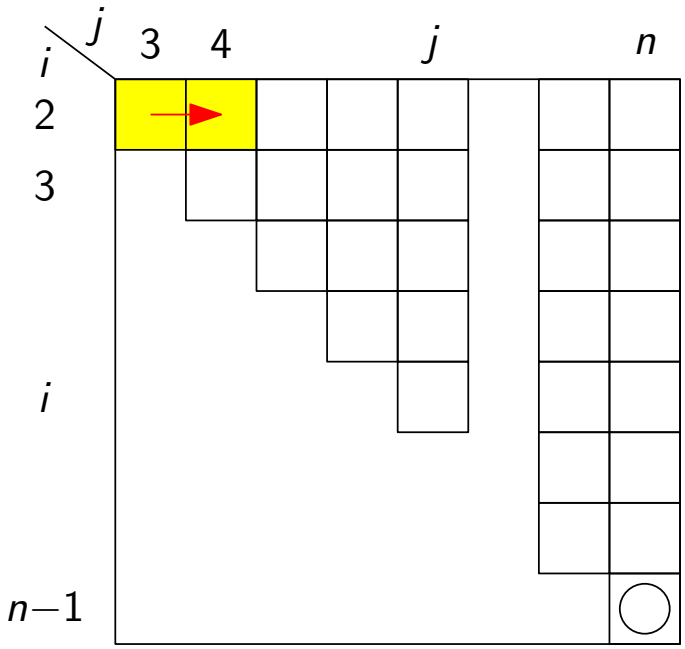



$$D(2,3) = c(1,2) + c(1,3)$$

# ピラミッド型巡回路近傍の探索：再帰式を解く (2)

## 再帰式 ($2 \leq i < j \leq n$)

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

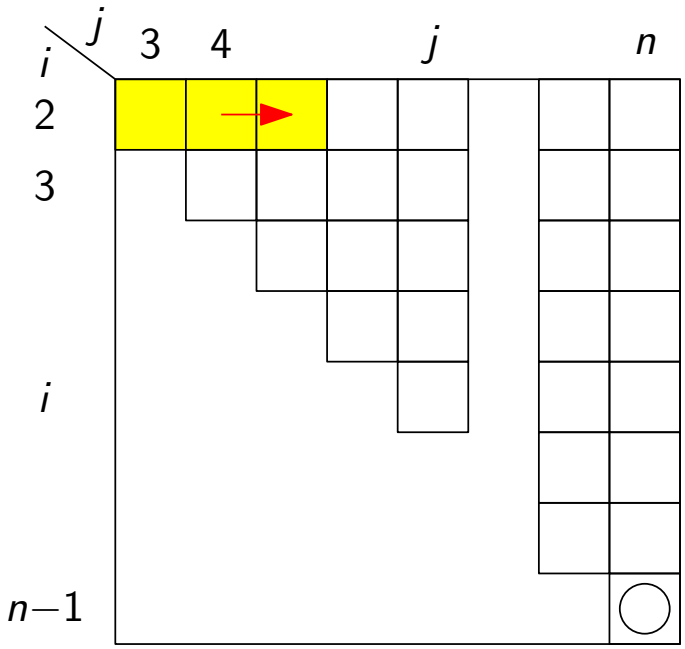



$$D(2,4) = D(2,3) + c(3,4)$$

# ピラミッド型巡回路近傍の探索：再帰式を解く (3)

## 再帰式 ($2 \le i < j \le n$)

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

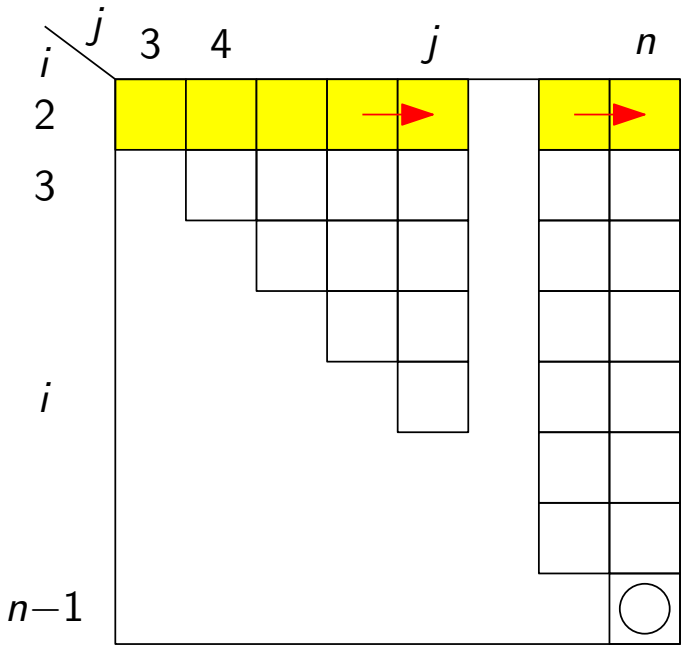



$$D(2,5) = D(2,4) + c(4,5)$$

# ピラミッド型巡回路近傍の探索：再帰式を解く (4)

## 再帰式 ($2 \leq i < j \leq n$)

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

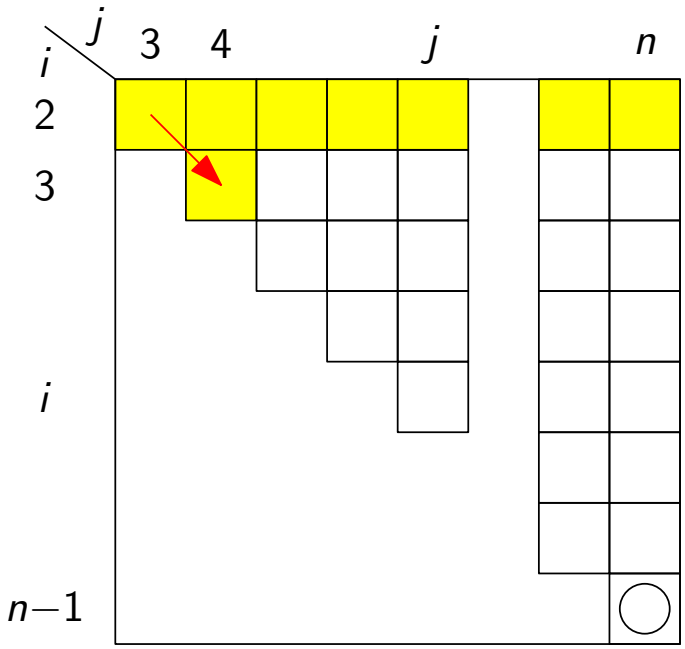



$$D(2,j) = D(2,j-1) + c(j-1,j)$$

# ピラミッド型巡回路近傍の探索：再帰式を解く (5)

## 再帰式 ($2 \leq i < j \leq n$)

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

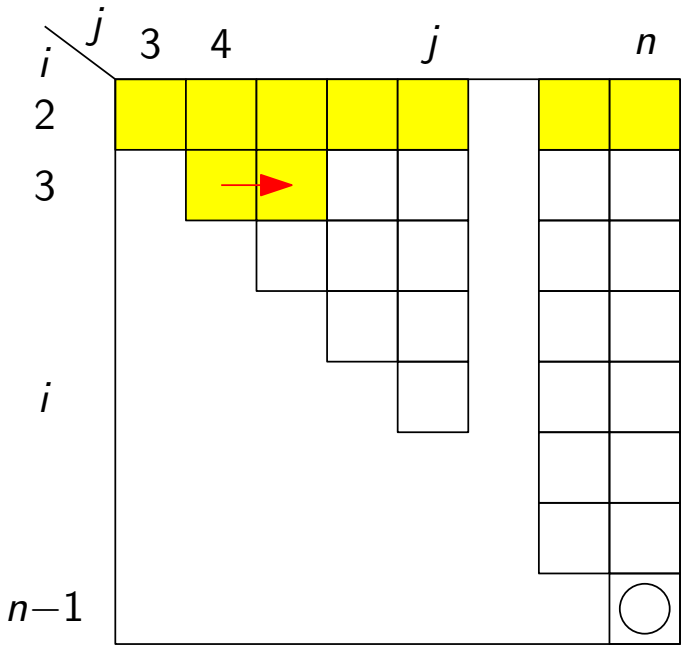



$$D(3,4) = D(2,3) + c(2,4)$$

# ピラミッド型巡回路近傍の探索：再帰式を解く (6)

## 再帰式 ($2 \leq i < j \leq n$)

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

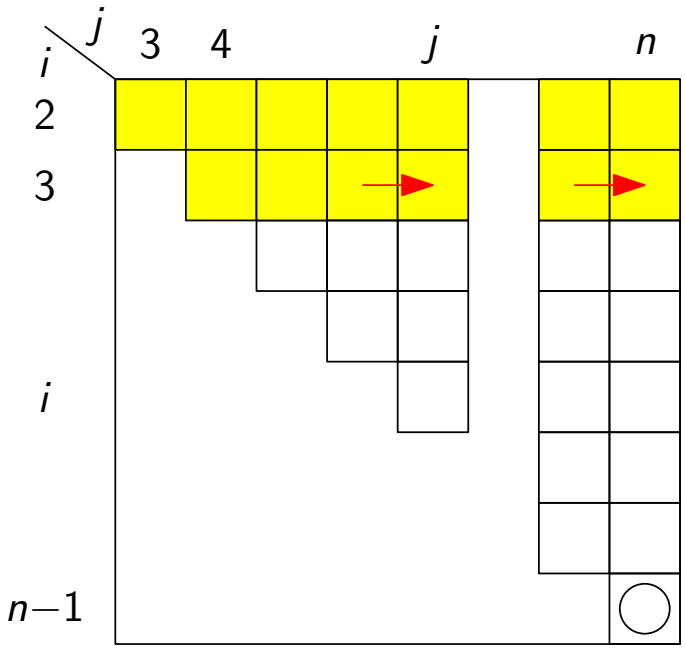



$$D(3,5) = D(3,4) + c(4,5)$$

# ピラミッド型巡回路近傍の探索：再帰式を解く (7)

## 再帰式 ($2 \leq i < j \leq n$)

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

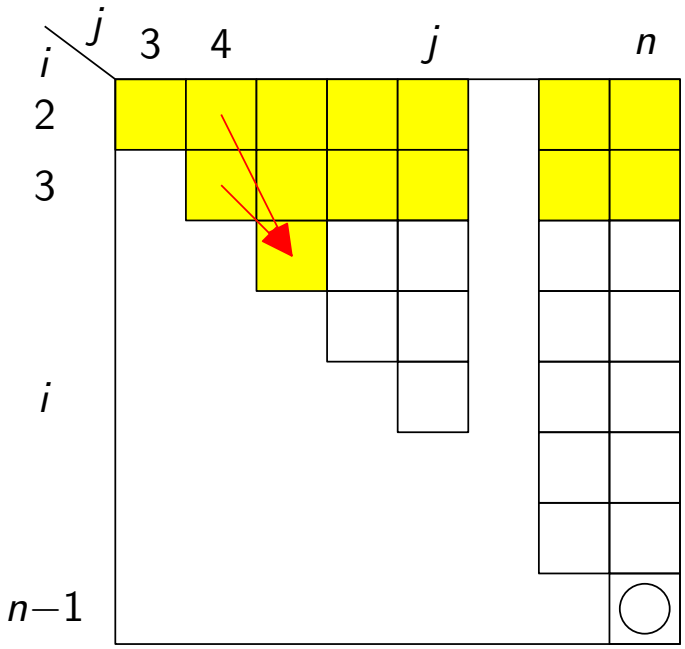



$$D(3,j) = D(3,j-1) + c(j-1,j)$$

# ピラミッド型巡回路近傍の探索：再帰式を解く (8)

## 再帰式 ($2 \le i < j \le n$)

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

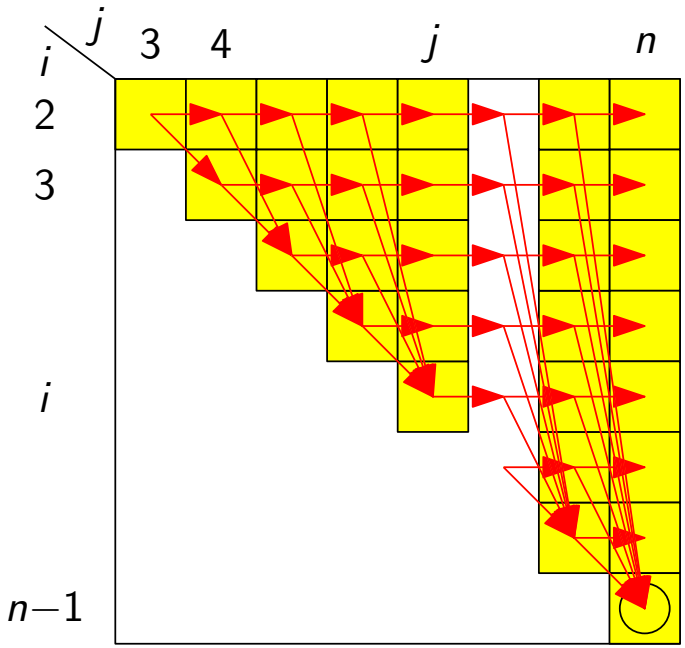



$$D(4,5) = \min\{D(3,4) + c(3,5), D(2,4) + c(2,5)\}$$

# ピラミッド型巡回路近傍の探索：再帰式を解く (9)

## 再帰式 ($2 \leq i < j \leq n$)

$$D(i,j) = \begin{cases} c(1,2) + c(1,3) & (i = 2,\ j = 3 \text{ のとき}) \\ D(i,j-1) + c(j-1,j) & (i+1 < j \text{ のとき}) \\ \min_{k<i}\{D(k,i) + c(k,j)\} & (i+1 = j \text{ のとき}) \end{cases}$$

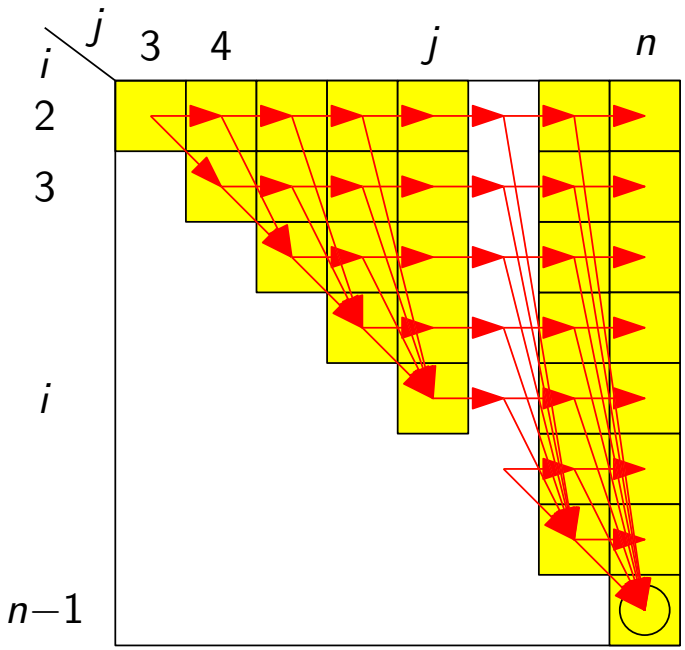


# ピラミッド型巡回路近傍の探索：再帰式を解く (詳細)

**常に，既に計算された $D(i,j)$ のみを使って計算していく**

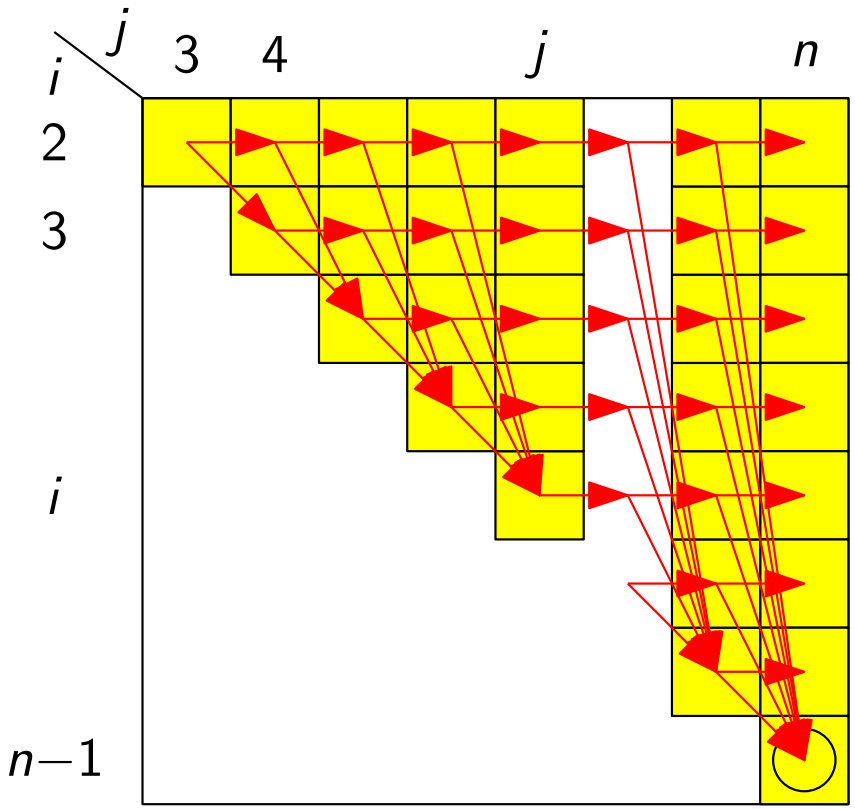
## 再帰式の解き方

0. $D(1,2) = c(1,2)$ (**計算の都合上，このように置く**)
1. **任意の** $i \in \{2,\ldots,n-1\}$ **に対して**
   1. $D(i,i+1) = \min_{k<i}\{D(k,i) + c(k,i+1)\}$
   2. **任意の** $j \in \{i+2,\ldots,n\}$ **に対して，** $D(i,j) = D(i,j-1) + c(j-1,j)$



# ピラミッド型巡回路近傍の探索：計算時間

## 再帰式を解くのにかかる時間 $\propto$ 赤い弧の数




- 赤い弧の数 $= O(n^2)$
- つまり，最短ピラミッド型巡回路は $O(n^2)$ 時間で見つかる □

# 目次

# 今回のまとめ

## いままでの感覚

**許容解 $x$ の近傍 $N(x)$ の中で，$x$ よりもよい許容解を見つけるためには，$|N(x)|$ に比例する時間は必要**

## 前回と今回のテーマ

**実はそうではなく，$|N(x)|$ よりも小さな時間で見つけられる (または，存在しないことが分かる)**

**今回の例 (超大規模近傍)**

- **巡回セールスマン問題の割当近傍** (Sarvanov, Doroshko '81)
  $O((n/2)!) \rightsquigarrow O(n^3)$
- **巡回セールスマン問題のピラミッド型巡回路近傍** (Carlier, Villon '90)
  $O(2^n) \rightsquigarrow O(n^2)$

**鍵となる概念**

- **効率よく解ける離散最適化問題**

# 残った時間の使い方

- 演習問題をやる
  - 相談推奨 (ひとりでやらない)
- 質問をする
  - 教員は巡回
- 退室時，小さな紙に感想など書いて提出する ← 重要
  - 内容は何でも OK
  - 匿名で OK

# 目次