数理経済学
第９回
ナップサック問題

塩浦昭義
東京工業大学 社会工学専攻 准教授
shioura.a.aa@m.titech.ac.jp

離散最適化問題

- 離散最適化問題とは:
  - 有限個の「もの」の組合せの中から，目的関数を最小または最大にする組合せを見つける問題
  - 例1:最小全域木，最短路，マッチング（グラフの枝の組合せ）
  - 例2:最小頂点被覆（グラフの頂点の組合せ）
  - 例3:ナップサック問題（「もの」の組合せ）
  - 例4:巡回セールスマン問題（都市の順列，枝の組合せ）

- 解きやすい問題と解きにくい問題
  - 解きやすい問題≒多項式時間で解ける問題
  - 解きにくい問題≒NP困難な問題
    （多項式時間で解けないと信じられている問題）

※離散最適化問題の解は有限個➔有限時間で必ず解ける！

ナップサック問題

ハイキングの準備

- n個の品物の中から持って行くものを選択
- ナップサックには b kg まで入れられる
- 品物 $i = 1,2,\ldots,n$ の重さは a_i kg, 価値は c_i
- 利用価値の合計を最大にしたい

最大化: $\sum_{i \in S} c_i\, x_i$

制約: $\sum_{i \in S} a_i\, x_i \leq b$

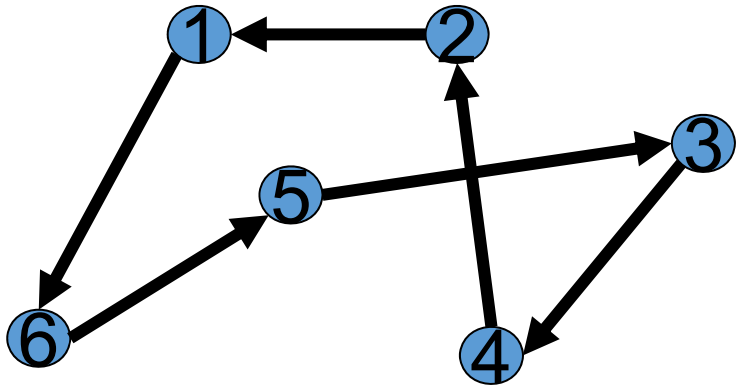
$S \subseteq \{1,2,\ldots,n\}$

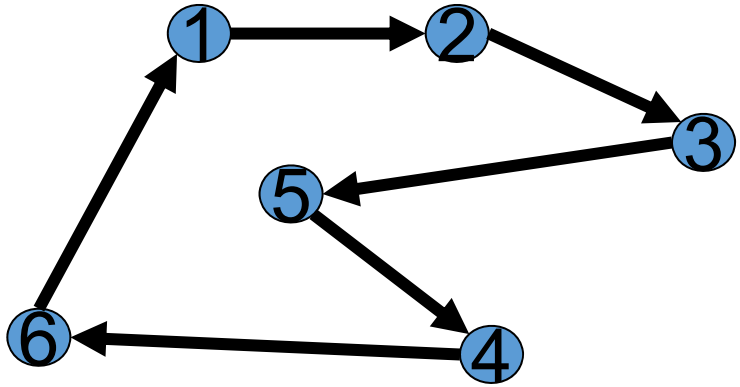
別の例: 財の購入

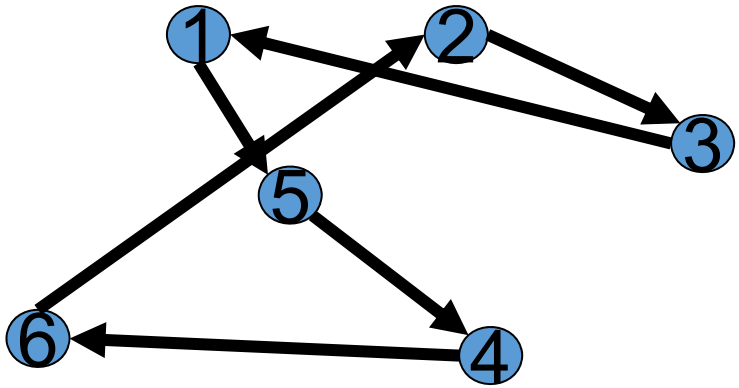
- n個の財の中から購入するものを選択
- 予算は b 円
- 財 $i = 1,2,\ldots,n$ の価格は a_i 円, 満足度は c_i
- 予算範囲内で満足度の合計を最大にしたい

巡回セールスマン問題

- セールスマンが n 都市をちょうど一回ずつ巡回する
- 都市 i から j への距離は c_{ij}（平面上の距離で与えられるケースも）
- 目的：都市を巡回する際の総距離を最小にする







離散最適化問題に対するアプローチ

どのように解くか？

- 解きやすい問題の場合
  - 多項式時間アルゴリズムを構築➔より高速な解法へ
- 解きにくい問題の場合
  - 絶対に最適解が必要➔厳密解法
    - 分枝限定法🡐現在の主流
    - 動的計画法
  - 高速に解を求めたい，ある程度良い解であれば十分
    - 精度保証付き近似アルゴリズム
      （解の良さに対する理論保証あり）
    - ヒューリスティックス（解の良さは実験的に証明）

今日の授業で紹介するアプローチ

ナップサック問題と連続版

ナップサック問題の定式化

最大化: $\sum_{i \in S} c_i\, x_i$
制約: $\sum_{i \in S} a_i\, x_i \leq b$
$S \subseteq \{1,2,\ldots,n\}$

$\Longleftrightarrow$

最大化: $\sum_{i=1}^{n} c_i\, x_i$
制約: $\sum_{i=1}^{n} a_i\, x_i \leq b$
$x_1, x_2, \ldots, x_n \in \{0,1\}$

$x_i = 1 \Longleftrightarrow i \in S$
$x_i = 0 \Longleftrightarrow i \notin S$

連続ナップサック問題: 変数の条件を$x_i \in \{0,1\}$から$0 \leq x_i \leq 1$へ

最大化: $\sum_{i=1}^{n} c_i\, x_i$
制約: $\sum_{i=1}^{n} a_i\, x_i \leq b$
$0 \leq x_i \leq 1 \;\; (i = 1,2,\ldots,n)$

連続ナップサック問題に対する貪欲アルゴリズム

- 資源（重量，金額）1単位あたりの価値（満足度）の高いものを選ぶ

貪欲アルゴリズム

ステップ1: n 個の品物を $\frac{c_{i_1}}{a_{i_1}} \geq \frac{c_{i_2}}{a_{i_2}} \geq \cdots \geq \frac{c_{i_n}}{a_{i_n}}$ を満たすように並べる

ステップ2: $a_{i_1} + a_{i_2} + \cdots + a_{i_{k-1}} \leq b < a_{i_k}$ を満たす k を求める

ステップ3: $x_{i_1} = x_{i_2} = \cdots = x_{i_{k-1}} = 1,$

$$x_{i_k} = \frac{b - \sum_{j=1}^{k-1} a_{i_j}}{a_{i_k}},$$

$x_{i_{k+1}} = x_{i_{k+2}} = \cdots = x_{i_n} = 0$ とする.

定理　貪欲アルゴリズムで得られた解は，
連続ナップサック問題の最適解

連続ナップサック問題に対する貪欲アルゴリズム：実行例

i	1	2	3	4	5	6	7	8
c_i	350	400	450	20	70	8	5	5
a_i	25	35	45	5	25	3	2	2

Martello, Toth: Knapsack Problems, Wiley (1990) より

b=104

c_i/a_i	14	11.4	10	8	2.8	2.7	2.5	2.5

25+35≦104<25+35+45

x_i	1	1	44/45	0	0	0	0	0

最適値＝1190

ナップサック問題に対する貪欲アルゴリズム

- 資源（重量，金額）1単位あたりの価値（満足度）の高いものを選ぶ

貪欲アルゴリズム

ステップ1: n 個の品物を $\frac{c_{i_1}}{a_{i_1}} \geq \frac{c_{i_2}}{a_{i_2}} \geq \cdots \geq \frac{c_{i_n}}{a_{i_n}}$ を満たすように並べる.

$S = \emptyset,\ j := 1$ とおく

ステップ2: $\sum_{i \in S} a_i + a_{i_j} \leq b$ ならば S に i_j を追加

ステップ3: $j = n$ ならば終了.

そうでなければ，j:=j+1 としてステップ2へ戻る

得られた解はどのくらい良い解か？

解の精度の評価

- 最適解（最適値）との比較で評価
- 定義： 近似比＝（得られた近似解の目的関数値）／最適値
  - 最大化問題の場合：近似比≦1，＝1ならば最適解
  - 最小化問題の場合：近似比≧1，＝1ならば最適解

- ナップサック問題の場合：近似比が1に近い解が欲しい

ナップサック問題に対する貪欲アルゴリズム:実行例

i	1	2	3	4	5	6	7	8
c_i	350	400	450	20	70	8	5	5
a_i	25	35	45	5	25	3	2	2

Martello, Toth: Knapsack Problems, Wiley (1990) より

b=104

c_i/a_i	14	11.4	10	8	2.8	2.7	2.5	2.5
	○	○	×	○	○	○	○	○
	79	44	44	39	14	11	9	7

x_i	1	1	0	1	1	1	1	1

目的関数値＝858　近似比＝0.9533

x_i^*	1	0	1	1	1	0	1	1

目的関数値＝900

ナップサック問題に対する貪欲アルゴリズム：悪い例

近似比が任意に悪くなる例が存在

i	1	2
c_i	1	b-1
a_i	1	b

bは任意の正整数

貪欲アルゴリズムの解＝(1,0), 目的関数値＝1

最適解＝(0,1), 目的関数値=b-1

貪欲アルゴリズムの解の近似比＝1/(b-1)
b→∞とすると近似比→0

ナップサック問題に対する貪欲アルゴリズムの改良

最初に容量オーバーになる品物（連続ナップサックのときのk）を利用

改良版貪欲アルゴリズム

ステップ1: n 個の品物を $\frac{c_{i_1}}{a_{i_1}} \geq \frac{c_{i_2}}{a_{i_2}} \geq \cdots \geq \frac{c_{i_n}}{a_{i_n}}$ を満たすように並べる.

$S = S' = \emptyset,\ j := 1$ とおく

ステップ2: $\sum_{i \in S} a_i + a_{i_j} \leq b$ ならば S に i_j を追加.

$\sum_{i \in S} a_i + a_{i_j} > b$ かつ$S' = \emptyset$ならば S' に i_j を追加.

ステップ3: $j = n$ ならば S と S' の良い方を出力して終了.

そうでなければ, j:=j+1 としてステップ2へ戻る

得られた解はどのくらい良い解か？

悪い例に対する
改良版貪欲アルゴリズムの適用

i	1	2
c_i	1	b-1
a_i	1	b

bは任意の正整数

改良版貪欲アルゴリズムの解　S＝(1,0), S’=(0,1),

よって，最適解が得られる

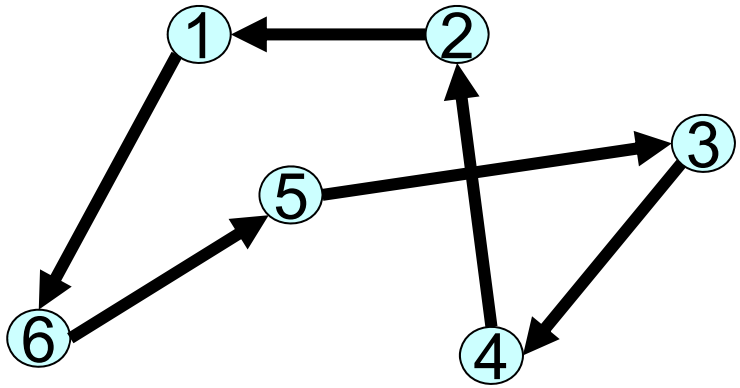
改良版貪欲アルゴリズムの近似比

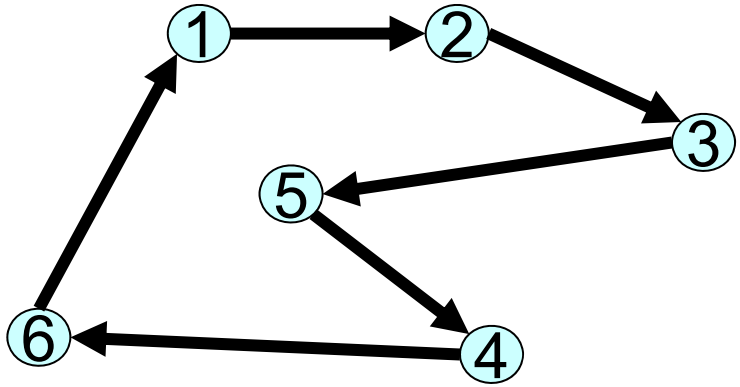
定理　ナップサック問題に対し，
　改良版貪欲アルゴリズムで得られた解の近似比は 1/2 以上

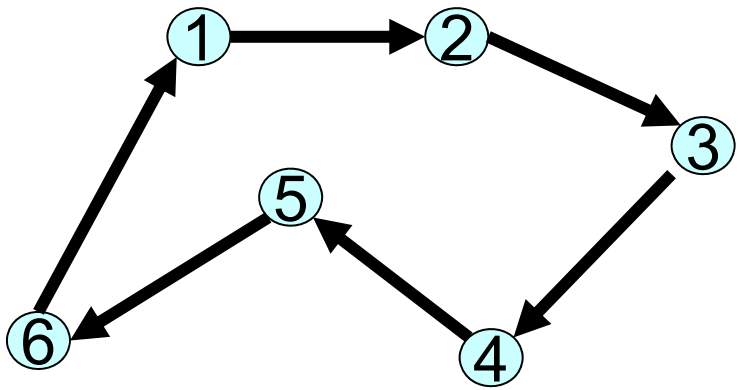
（証明）連続ナップサック問題の方が解の選択肢が多いので，
　ナップサック問題の最適値≦連続ナップサック問題の最適値
連続ナップサック問題の最適解：{1,2,…,k-1} は全部，{k} は一部分
改良版貪欲アルゴリズムにおいて {1,2,…,k-1}⊆S,　k∈S’
➔（Sの目的関数値）＋（S’の目的関数値）
　　≧連続ナップサック問題の最適値
∴改良版貪欲アルゴリズムの解
≧(1/2){（Sの目的関数値）＋（S’の目的関数値）}
≧(1/2)連続ナップサック問題の最適値
≧(1/2)ナップサック問題の最適値

巡回セールスマン問題

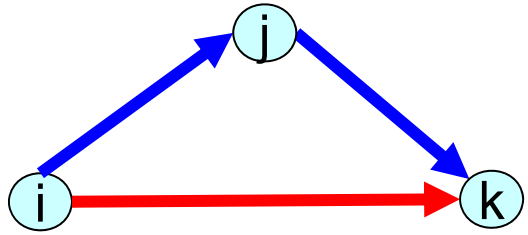
- 入力: n 個の点 1,2,…,n, および各点間の距離 d(i,j)
- 目的: 1から始まる点の順列$i_1 = 1, i_2, \dots, i_n$
  のうち，値$\sum_{j=1}^{n-1} d\left(i_j, i_{j+1}\right) + d(i_n, i_1)$ が最小なもの







距離に関する仮定



- 距離は非負: d(i,j)≧0 (∀i,j)
- 対称: d(i,j)=d(j,i) (∀i,j)
- 三角不等式を満たす: d(i,k)≦d(i,j)+d(j,k) (∀i,j,j)

仮定を満たす距離の例:

- ユークリッド距離(ℓ_2距離): $\sqrt{\sum_i (x_i - y_i)^2}$
- マンハッタン距離(ℓ_1距離): $\sum_i |x_i - y_i|$
- チェビシェフ距離(ℓ_0距離): $\max_i |x_i - y_i|$
- 文字列に対するハミング距離: 2つの文字列の距離＝異なる文字数

  d(make, cakes)=2,　d(shimoda, shioura)=3

など多数

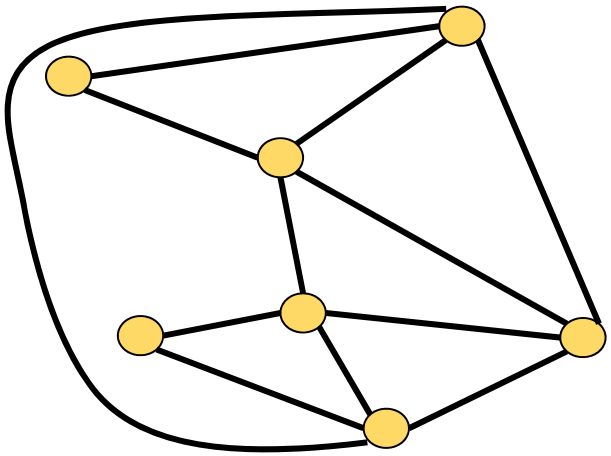
2種類の近似アルゴリズム

- 最小全域木を用いた2近似アルゴリズム
- 最小全域木＋最小重みマッチングを用いた1.5近似アルゴリズム

準備：アルゴリズムでは以下の性質を利用

定理　無向グラフにおいて，各頂点の次数が偶数ならば，
全ての枝を一筆書きでたどることができる（オイラー閉路とよぶ）

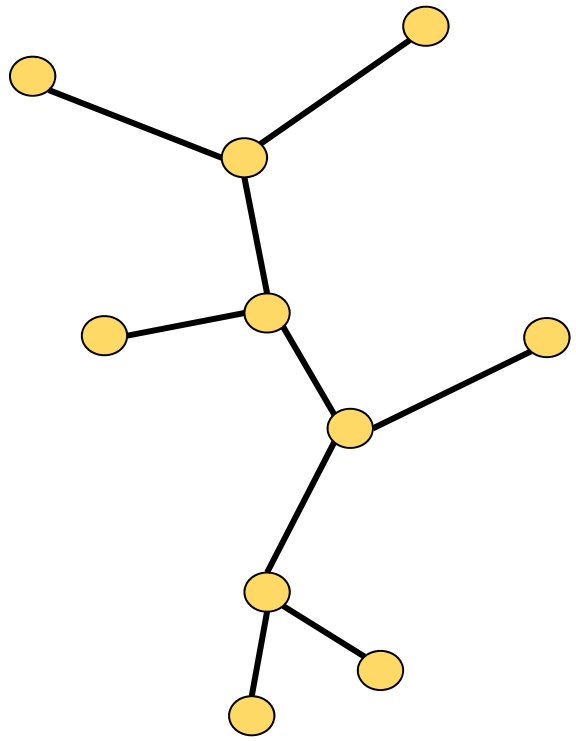
（雑な証明）適当に枝をたどって
いけば，必ず一筆書きできる
（なぜ？）



最小全域木を用いた2近似アルゴリズム

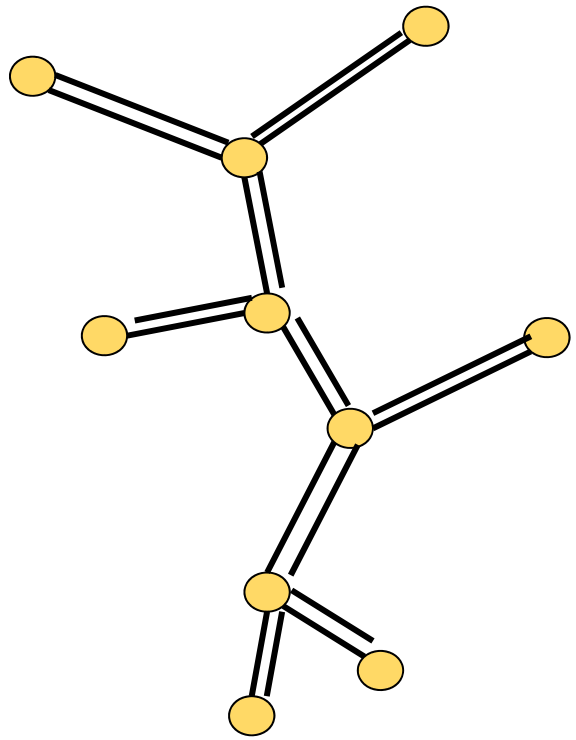
最小全域木を用いた2近似アルゴリズム

ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)
ステップ2: 最小全域木の枝を全て二重にする.
ステップ3: オイラー閉路を作る.
ステップ4: オイラー閉路の無駄なところをショートカットして，巡回路にする.



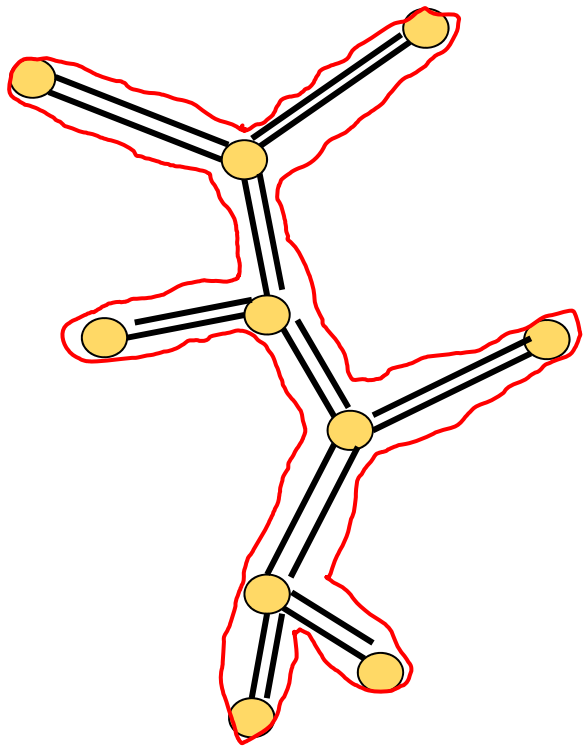
最小全域木を用いた2近似アルゴリズム

ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)
ステップ2: 最小全域木の枝を全て二重にする.
ステップ3: オイラー閉路を作る.
ステップ4: オイラー閉路の無駄なところをショートカットして，巡回路にする.



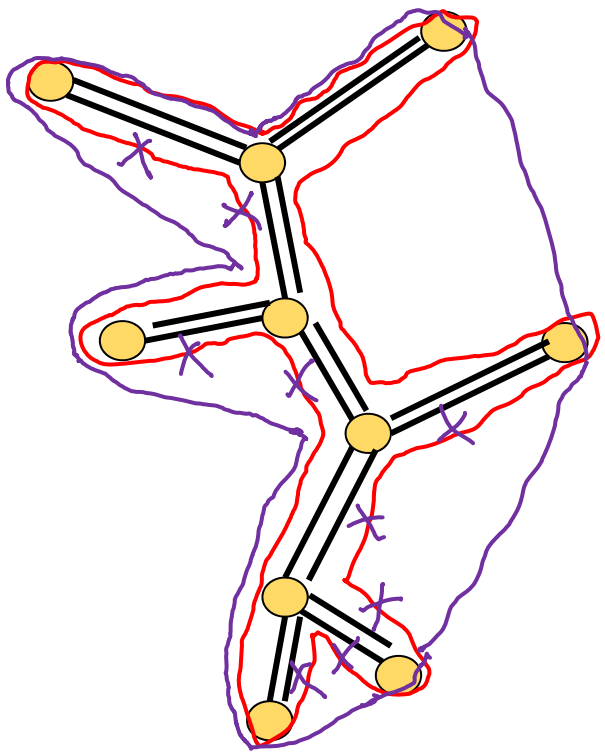
最小全域木を用いた2近似アルゴリズム

ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)
ステップ2: 最小全域木の枝を全て二重にする.
ステップ3: オイラー閉路を作る.
ステップ4: オイラー閉路の無駄なところをショートカットして，巡回路にする.



最小全域木を用いた2近似アルゴリズム

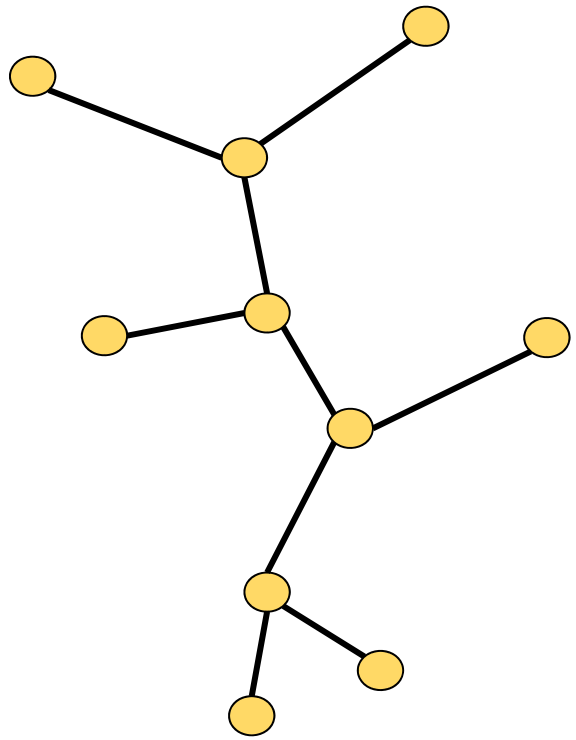
ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)
ステップ2: 最小全域木の枝を全て二重にする.
ステップ3: オイラー閉路を作る.
ステップ4: オイラー閉路の無駄なところをショートカットして，巡回路にする.



アルゴリズムの近似比解析

ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)

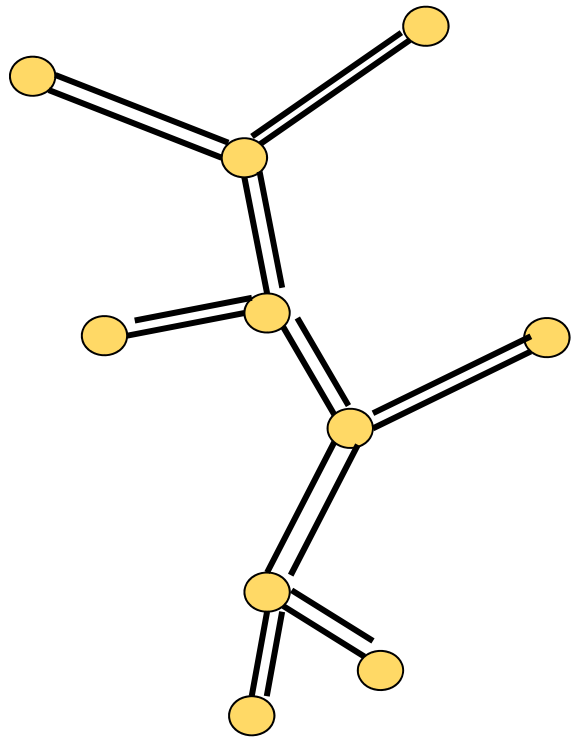
最小全域木の総距離≦最短巡回路長
(∵巡回路から枝をひとつ取り除くと
全域木)



アルゴリズムの近似比解析

ステップ2: 最小全域木の枝を全て二重にする.

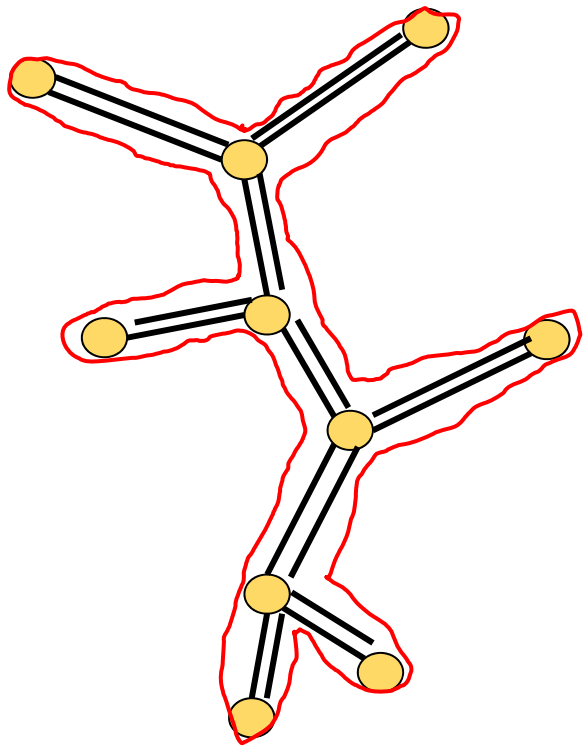
枝の長さの合計
≦2×(最小全域木の長さ)



アルゴリズムの近似比解析

ステップ3: オイラー閉路を作る.

オイラー閉路の長さ＝枝の長さの合計

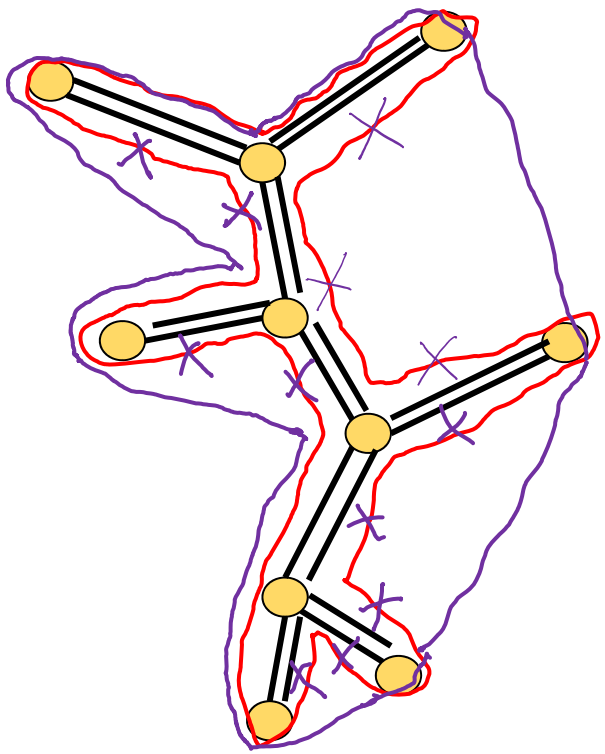


アルゴリズムの近似比解析

ステップ4：オイラー閉路の無駄なところをショートカットして，巡回路にする．

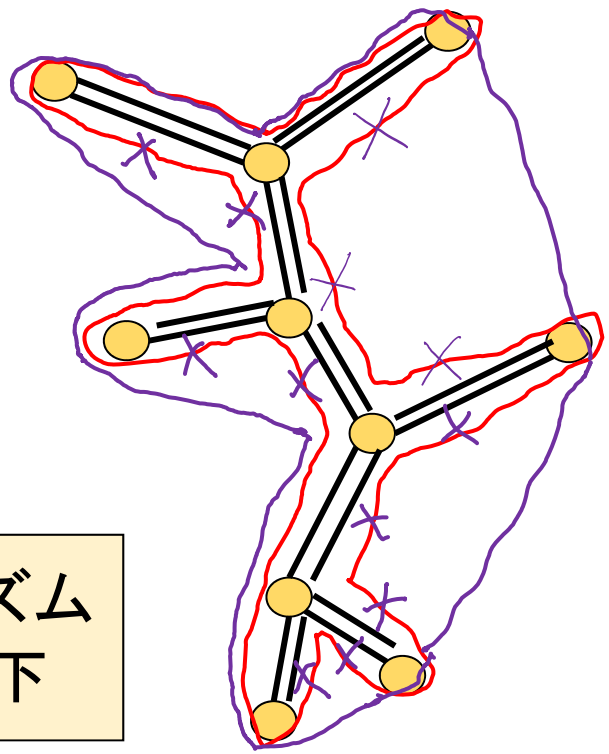
三角不等式により，
ショートカットの長さ
　　≦ショートカットされた部分の長さ

よって，
得られた巡回路の長さ
　　≦オイラー閉路の長さ



アルゴリズムの近似比解析：まとめ

得られた巡回路の長さ
≦オイラー閉路の長さ
＝2×（最小全域木の長さ）
≦ 2×（最短巡回路の長さ）



定理　最小全域木を用いた近似アルゴリズムの近似比は2以下

最小全域木と最小重みマッチングを用いた1.5近似アルゴリズム

最小全域木と最小重みマッチングを用いた1.5近似アルゴリズム：準備

最小重みkマッチング問題：

　枝数kのマッチングで枝重みの和が最小のものを求める

最小重み完全マッチング問題：

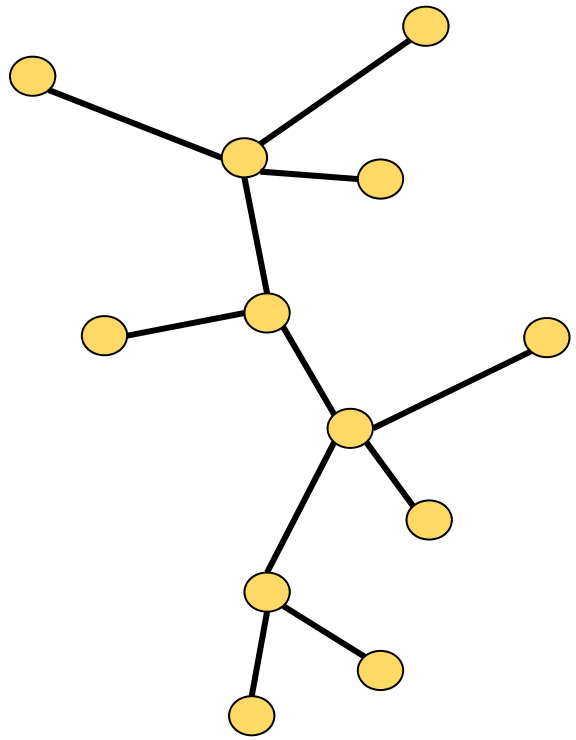
　全ての頂点を繋ぐマッチングで枝重みの和が最小のものを求める

最小重みk（完全）マッチング問題の解き方：

枝重みに -1 を掛けて，最大重みk（完全）マッチング問題を解けばよい

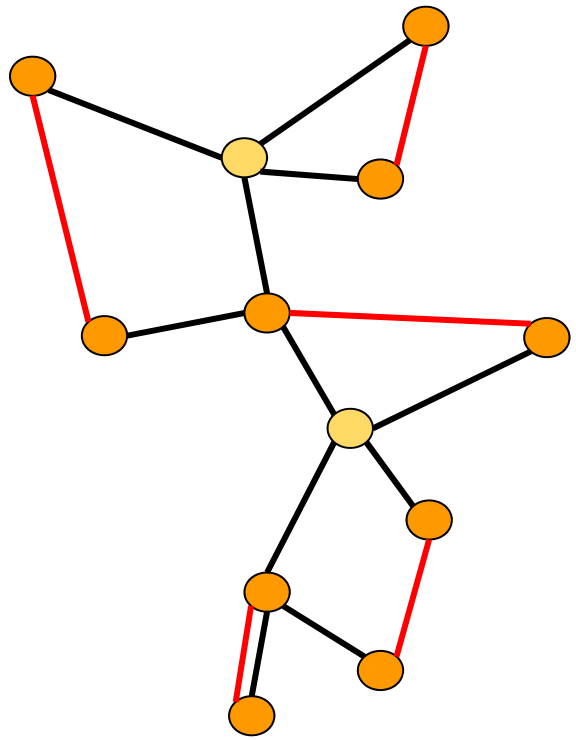
最小全域木と最小重みマッチングを用いた1.5近似アルゴリズム

ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)
ステップ2: 奇数次数の頂点のみに関する最小重み完全マッチングを求める
ステップ3: オイラー閉路を作る.
ステップ4: オイラー閉路の無駄なところをショートカットして，巡回路にする.



最小全域木と最小重みマッチングを用いた1.5近似アルゴリズム

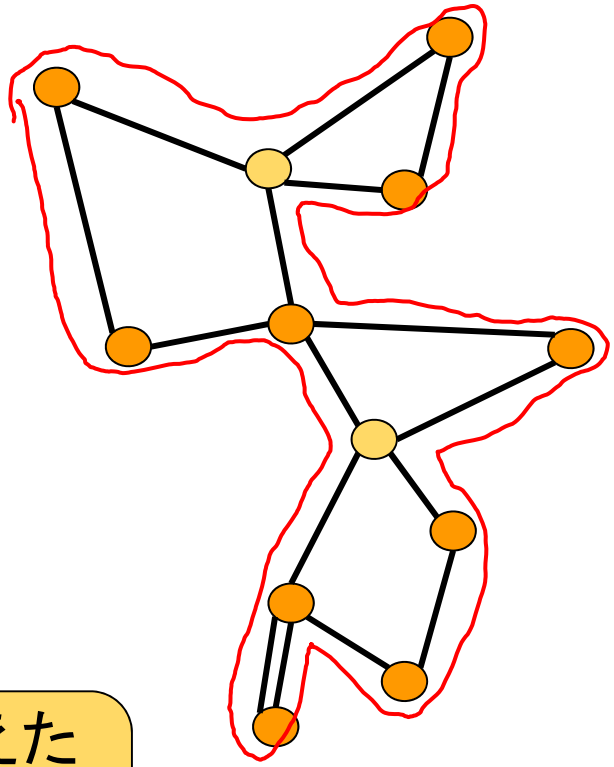
ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)
ステップ2: 奇数次数の頂点のみに関する最小重み完全マッチングを求める
ステップ3: オイラー閉路を作る.
ステップ4: オイラー閉路の無駄なところをショートカットして，巡回路にする.



奇数次数の頂点の総数は偶数
➔完全マッチングが存在する

最小全域木と最小重みマッチングを用いた1.5近似アルゴリズム

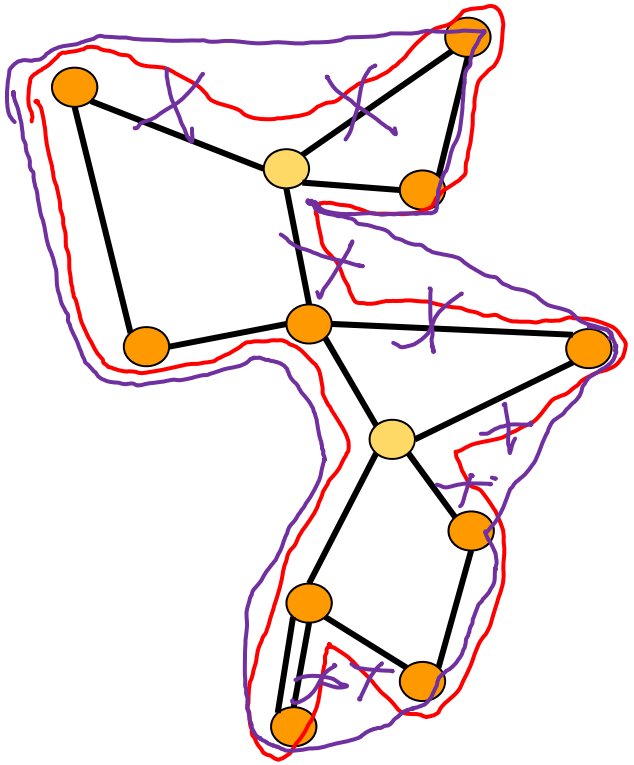
ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)
ステップ2: 奇数次数の頂点のみに関する最小重み完全マッチングを求める
ステップ3: オイラー閉路を作る.
ステップ4: オイラー閉路の無駄なところをショートカットして，巡回路にする.



奇数次数の頂点の次数はいずれも1増えた
➔全ての頂点の次数が偶数
➔オイラー閉路が存在

最小全域木と最小重みマッチングを用いた1.5近似アルゴリズム

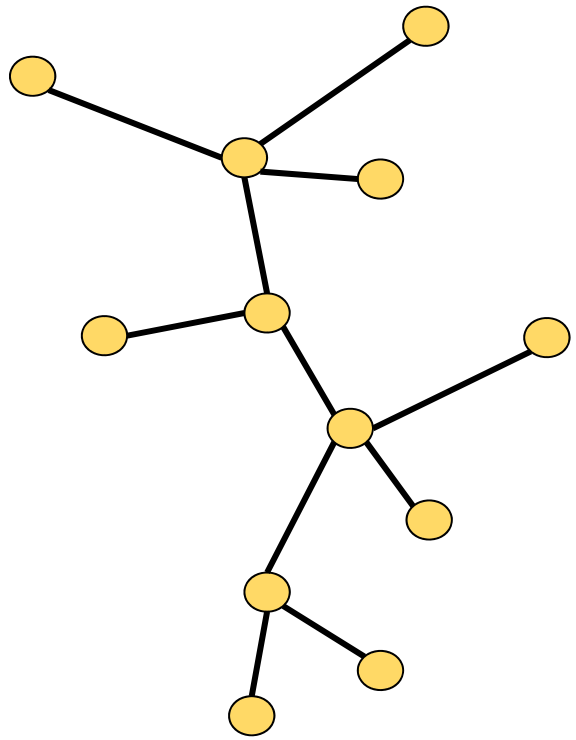
ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)
ステップ2: 奇数次数の頂点のみに関する最小重み完全マッチングを求める
ステップ3: オイラー閉路を作る.
ステップ4: オイラー閉路の無駄なところをショートカットして，巡回路にする.



アルゴリズムの近似比解析

ステップ1: 最小全域木を求める
ただし，次のグラフを考える.
V=全ての点集合
E=全ての異なる点の対
枝(i,j)の費用=d(i,j)

最小全域木の総距離≦最短巡回路長
(∵巡回路から枝をひとつ取り除くと
全域木)

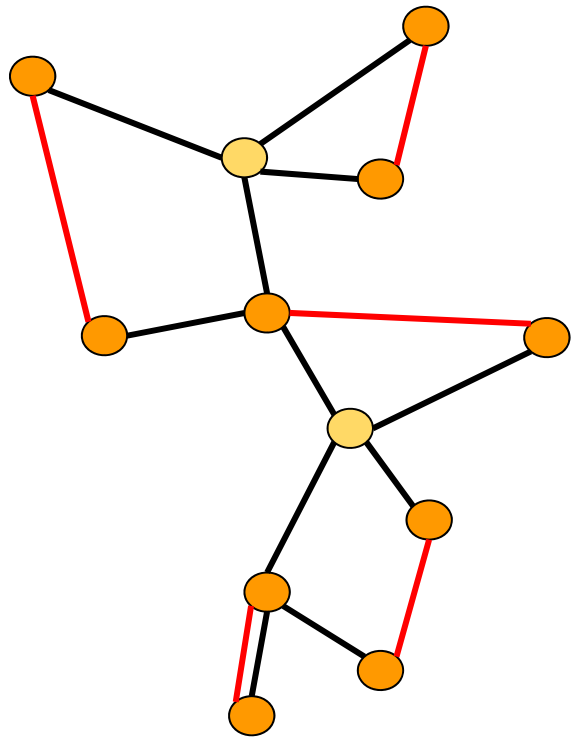


アルゴリズムの近似比解析

ステップ2：奇数次数の頂点のみに
関する最小重み完全マッチングを求める

最小重み完全マッチングの総距離
≦0.5×最短巡回路長

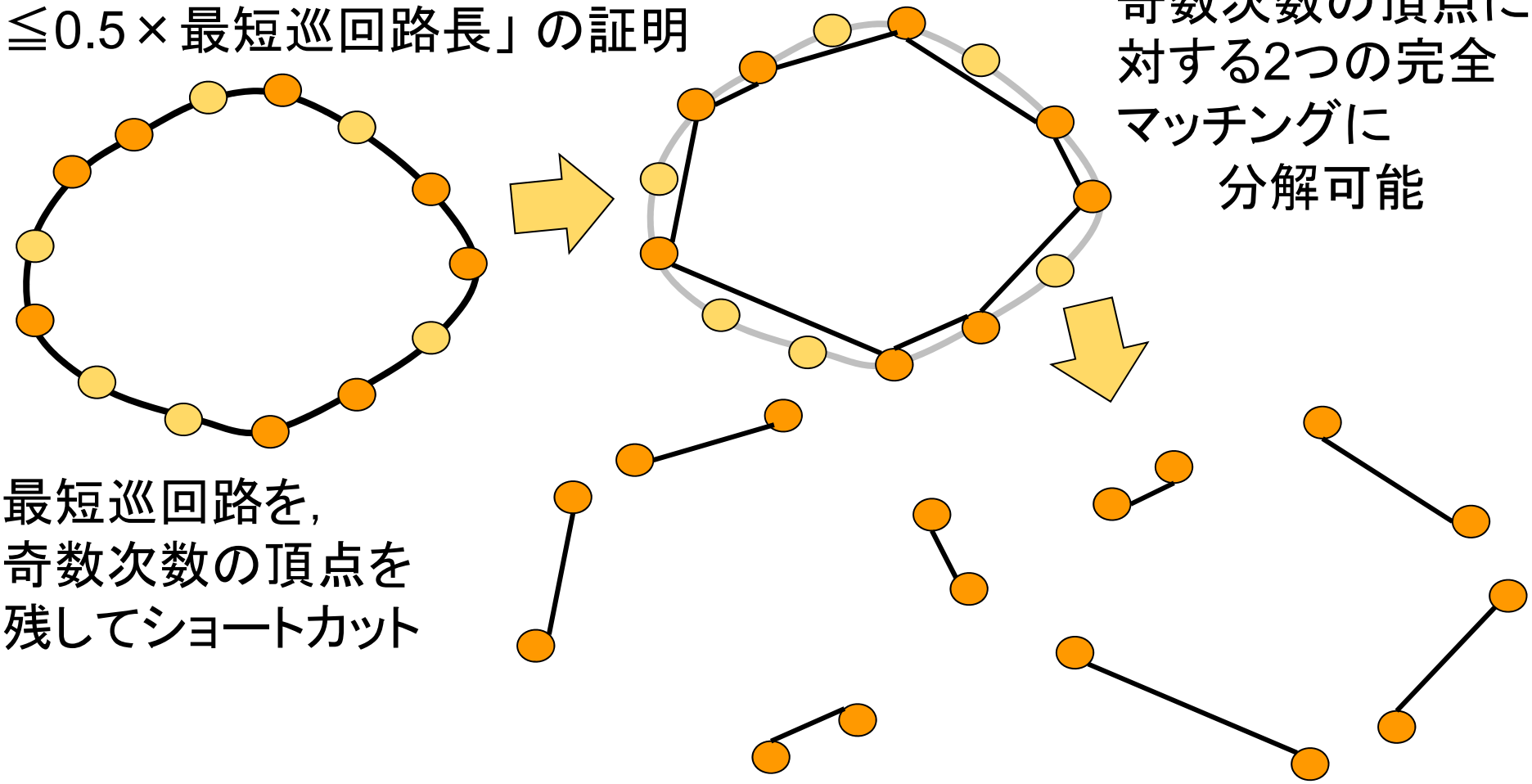
（証明）
最短巡回路長
≧奇数次数の頂点を残してショートカット
　して得られた閉路の長さ
＝奇数次数の頂点に対する2つの
　完全マッチングの総距離の和
≧ ２×最小重みマッチングの総距離



アルゴリズムの近似比解析

「最小重み完全マッチングの総距離
≦0.5×最短巡回路長」の証明

奇数次数の頂点に
対する2つの完全
マッチングに
分解可能



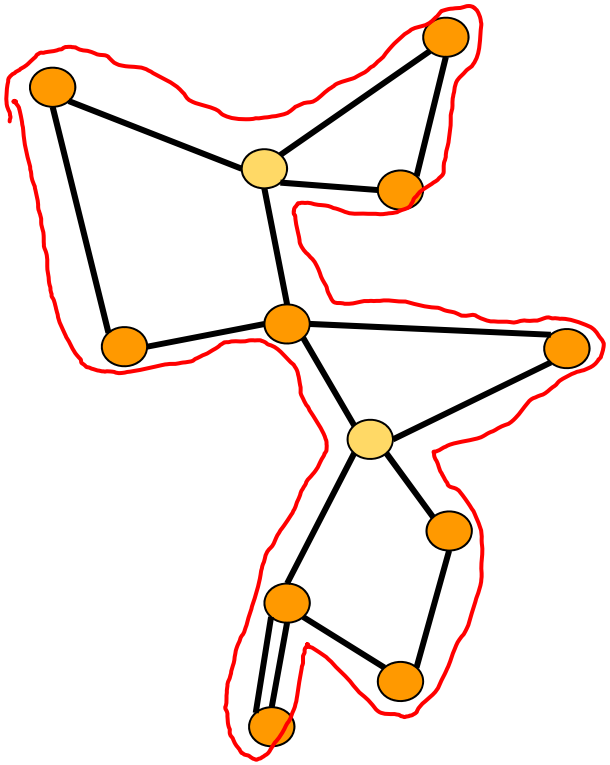
最短巡回路を,
奇数次数の頂点を
残してショートカット

各完全マッチングの総距離≧最小重み完全マッチングの総距離

アルゴリズムの近似比解析

ステップ3: オイラー閉路を作る.

オイラー閉路の長さ
＝最小全域木の総距離
　＋ 最小重みマッチングの総距離

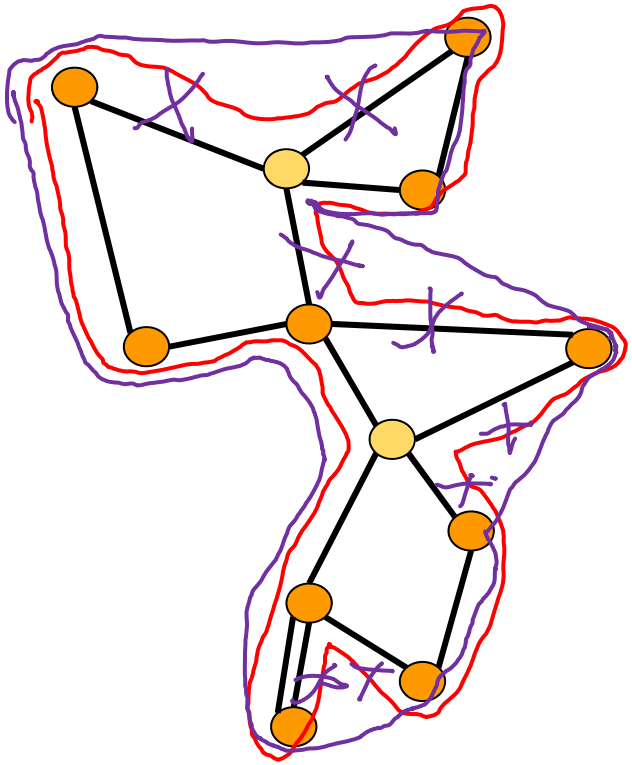


アルゴリズムの近似比解析

ステップ4：オイラー閉路の無駄なところをショートカットして，巡回路にする．

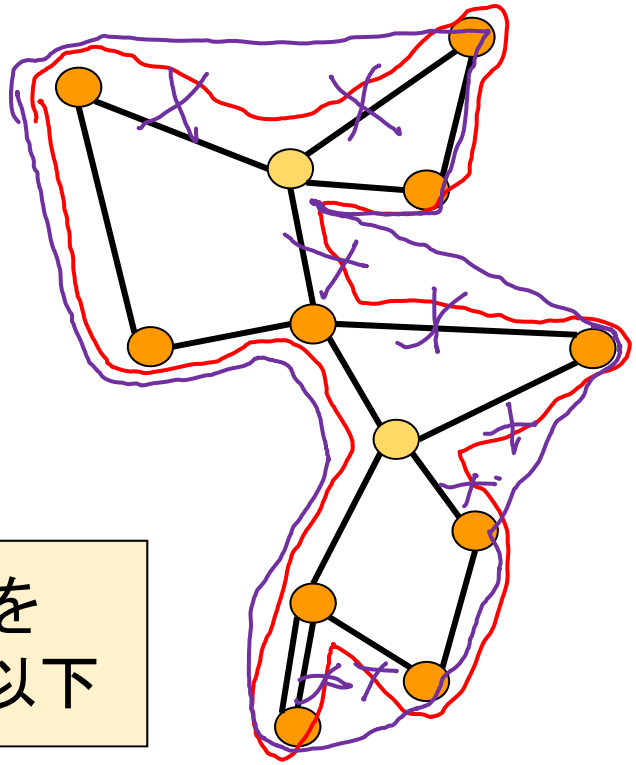
三角不等式により，
ショートカットの長さ
　　≦ショートカットされた部分の長さ

よって，
得られた巡回路の長さ
　　≦オイラー閉路の長さ



アルゴリズムの近似比解析：まとめ

得られた巡回路の長さ
≦オイラー閉路の長さ
＝最小全域木の総距離
　　＋最小重みマッチングの総距離
≦（最短巡回路の長さ）
　　＋0.5×（最短巡回路の長さ）
＝1.5×（最短巡回路の長さ）



定理　最小全域木と最小重みマッチングを用いた近似アルゴリズムの近似比は1.5以下

演習問題

問1 : 下記のナップサック問題の例に対し，改良版貪欲アルゴリズムを適用して近似解を計算せよ.

i	1	2	3	4	5	6
c_i	10	7	25	24	28	10
a_i	2	1	6	5	7	3

b=7

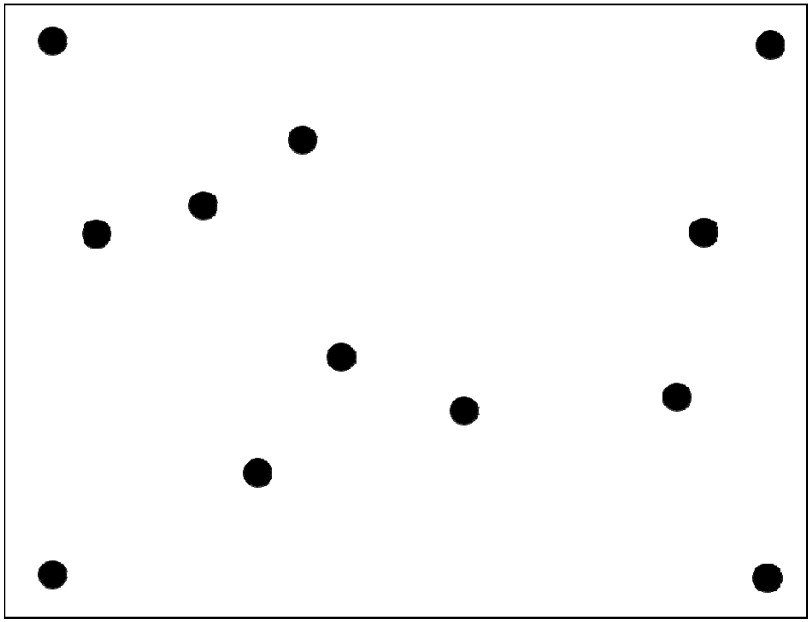
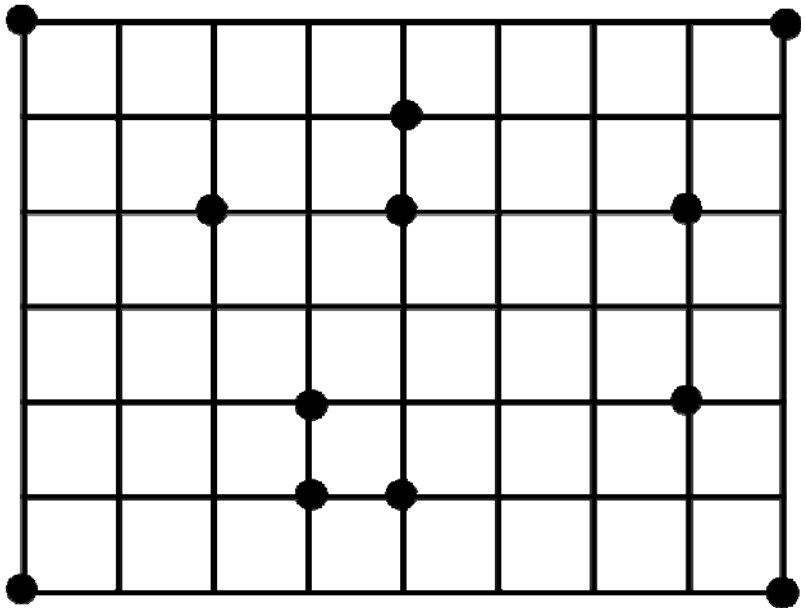
問2 : 上記の問題例を連続ナップサック問題と見なしたときの最適解を貪欲アルゴリズムを使って計算せよ.

問3: 改良版貪欲アルゴリズムの解の近似比がほぼ 1/2 になる例を示しなさい.

演習問題

問4: 以下の巡回セールスマン問題の問題例に対して,
2近似アルゴリズムおよび1.5近似アルゴリズムを適用して得られる解を求めよ. ただし, 最小重みマッチングの計算については厳密でなくても可. (1) については, 最小全域木の計算も厳密でなくても可.

(1) 点の間の長さは
ユークリッド距離とする



(2) 点の間の長さは
マンハッタン距離とする