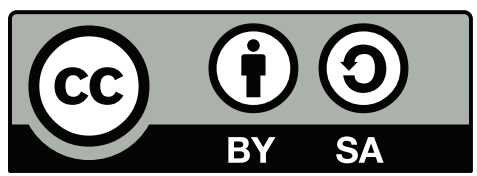
オペレーションズ・リサーチ III (1)

田中 俊二

shunji.tanaka@okayama-u.ac.jp



スケジュール

No.	内容
1	導入 (組合せ最適化，グラフ・ネットワーク，整数計画問題)
2	計算複雑さの理論
3	グラフ・ネットワーク 1 (グラフの分類，用語，種々の問題)
4	グラフ・ネットワーク 2 (最短経路問題，動的計画法)
5	グラフ・ネットワーク 3 (最小全域木，最大フロー問題)
6	グラフ・ネットワーク 4 (マッチング)
7	整数計画 (緩和問題，分枝限定法，切除平面法)

オペレーションズ・リサーチ III の内容

- 組合せ最適化問題 (combinatorial optimization problem)
- 計算複雑さの理論 (computational complexity)
- グラフ・ネットワーク (graph and network)
- 整数計画問題 (integer programming problem)

組合せ最適化問題 (combinatorial optimization problem)

- 解が組合せで表されるような最適化問題
- 順列 (permutation)，割り当て (assignment)，etc.
- **グラフ・ネットワーク**上の問題や**整数計画問題**として扱える

整数計画問題 (integer programming problem)

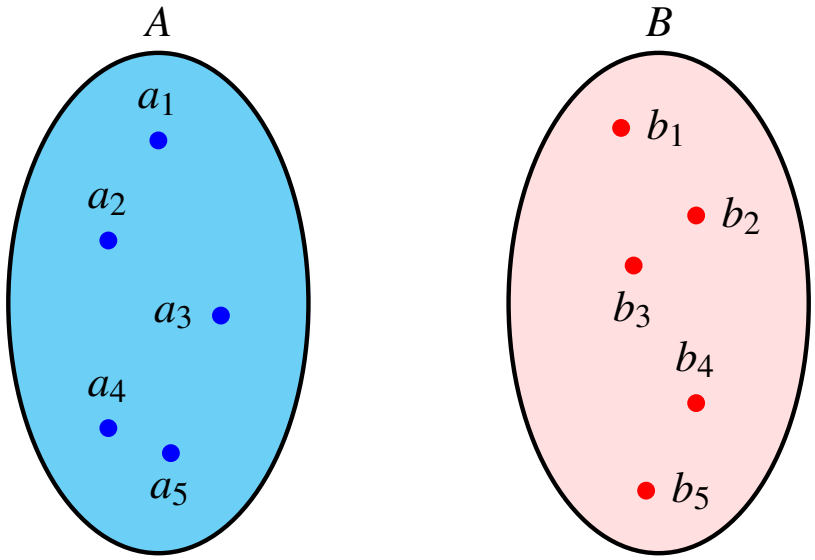
- 決定変数が整数値のみを取る最適化問題

離散最適化問題 (discrete optimization problem)

- 実行可能領域が離散的 (飛び飛びの値)
- 実行可能解が無限個の場合も含む (組合せ最適化問題は有限個)
- 組合せ最適化問題と同じ意味で使われることがほとんど

組合せ最適化問題：(線形) 割当問題

同サイズの 2 つの集合間の 1 対 1 関係 (割り当て) を最適化する問題

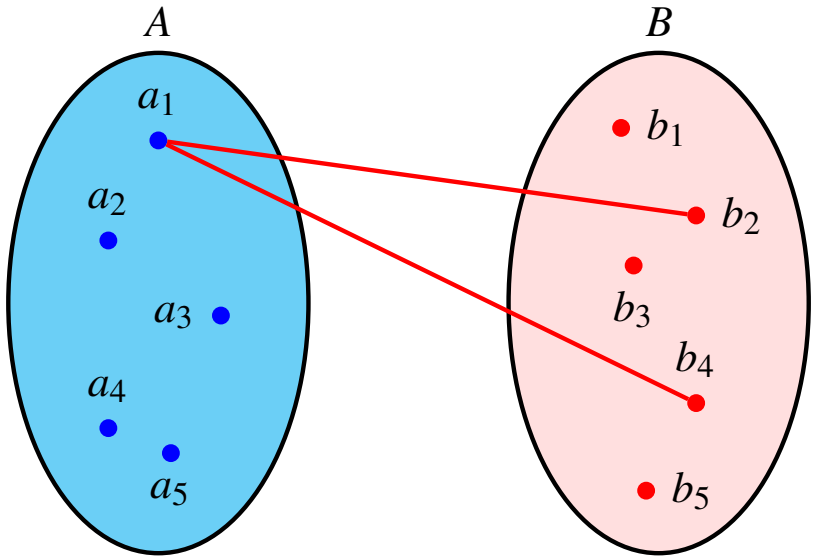


(線形) 割当問題 ((linear) assignment problem)

- 集合 A の n 個の要素 $a_1, \ldots, a_n$ を，集合 B の n 個の要素 $b_1, \ldots, b_n$ に一つずつ割り当てる
- a_i を b_j に割り当てたときのコスト: c_{ij}
- 同じ a_i を 2 つ以上の b_j に割り当てる... NG
- 同じ b_j に 2 つ以上の a_i を割り当てる... NG
- 総コストを最小化

組合せ最適化問題：(線形) 割当問題

同サイズの 2 つの集合間の 1 対 1 関係 (割り当て) を最適化する問題

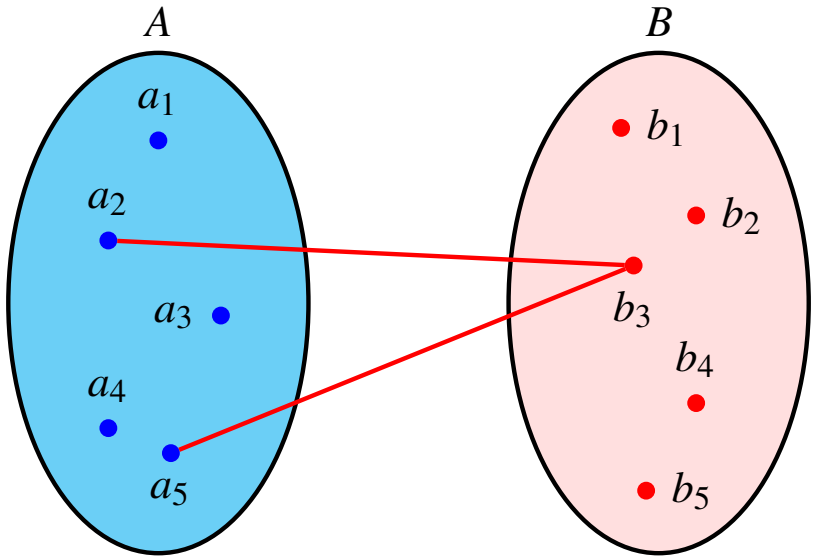


(線形) 割当問題 ((linear) assignment problem)

- 集合 A の n 個の要素 $a_1, \ldots, a_n$ を，集合 B の n 個の要素 $b_1, \ldots, b_n$ に一つずつ割り当てる
- a_i を b_j に割り当てたときのコスト: c_{ij}
- **同じ a_i を 2 つ以上の b_j に割り当てる... NG**
- 同じ b_j に 2 つ以上の a_i を割り当てる... NG
- 総コストを最小化

組合せ最適化問題：(線形) 割当問題

同サイズの 2 つの集合間の 1 対 1 関係 (割り当て) を最適化する問題

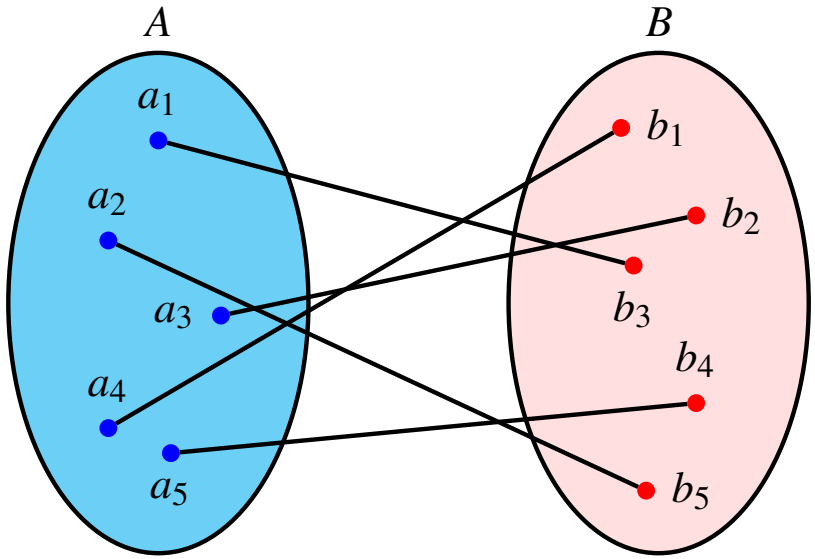


(線形) 割当問題 ((linear) assignment problem)

- 集合 A の n 個の要素 $a_1, \ldots, a_n$ を，集合 B の n 個の要素 $b_1, \ldots, b_n$ に一つずつ割り当てる
- a_i を b_j に割り当てたときのコスト: c_{ij}
- 同じ a_i を 2 つ以上の b_j に割り当てる... NG
- **同じ b_j に 2 つ以上の a_i を割り当てる... NG**
- 総コストを最小化

組合せ最適化問題：(線形) 割当問題

同サイズの 2 つの集合間の 1 対 1 関係 (割り当て) を最適化する問題



(線形) 割当問題 ((linear) assignment problem)

- 集合 A の n 個の要素 $a_1, \ldots, a_n$ を，集合 B の n 個の要素 $b_1, \ldots, b_n$ に一つずつ割り当てる
- a_i を b_j に割り当てたときのコスト: c_{ij}
- 同じ a_i を 2 つ以上の b_j に割り当てる... NG
- 同じ b_j に 2 つ以上の a_i を割り当てる... NG
- 総コストを最小化

割当問題の例

- 3 人のアルバイト X, Y, Z を午前, 午後, 深夜のシフトに一人ずつ割り当てる
  $A = \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$, $B = \{$ 午前, 午後, 深夜 $\}$
- 割り当てのコスト: 表の通り
- 総コスト最小の勤務表を作成

割り当てコスト

バイト \ シフト	午前	午後	深夜
X	5	6	7
Y	4	5	8
Z	6	4	9

割当問題の例

- 3 人のアルバイト X, Y, Z を午前, 午後, 深夜のシフトに一人ずつ割り当てる
  $A = \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$, $B = \{$ 午前, 午後, 深夜 $\}$
- 割り当てのコスト: 表の通り
- 総コスト最小の勤務表を作成

割り当てコスト

バイト \ シフト	午前	午後	深夜
X	5	6	**7**
Y	**4**	5	8
Z	6	**4**	9

最適解

- X: 深夜, Y: 午前, Z: 午後
- 最適値 $7 + 4 + 4 = 15$

組合せ最適化問題：ナップサック問題

- ナップサックにアイテムを詰めていく問題

0-1 ナップサック問題 (0-1 knapsack problem)

- n 個のアイテム
- アイテム i の情報
  - サイズ (size) a_i
  - 利得 (profit) c_i
- ナップサックの容量 (capacity) C
- サイズの総和が容量 C を超えない範囲で各アイテムを取捨選択し，利得の総和を最大化
- 同じアイテムを**重複して選択することはできない**

無制限ナップサック問題 (unbounded knapsack problem)

- 設定は 0-1 ナップサック問題と同じ
- ただし，同じアイテムを**重複して選択してもよい**

ナップサック問題の例

- 表のメニューから選ぶ
- 総カロリーを 1900 kcal 以下に抑えつつ，タンパク質の総量を最大化
  - アイテムのサイズ：カロリー
  - 利得：タンパク質
  - ナップサック容量：総カロリー

マクドナルドのメニューの栄養情報 †

種類	カロリー (kcal)	タンパク質 (g)
ハンバーガー	256	12.8
チーズバーガー	307	15.7
ビッグマック	530	26.4
フィレオフィッシュ	336	14.6
てりたまバーガー	566	21.8
チキンフィレオ	470	20.2
ベーコンレタスバーガー	374	18.1
チキンマックナゲット 5 ピース	263	15.3
えだまめコーン	83	5.2

† https://www.mcdonalds.co.jp/quality/allergy_Nutrition/nutrient/

ナップサック問題の例 (続き)

マクドナルドのメニューの栄養情報

種類	カロリー (kcal)	タンパク質 (g)
ハンバーガー	256	12.8
チーズバーガー	307	15.7
ビッグマック	530	26.4
フィレオフィッシュ	336	14.6
てりたまバーガー	566	21.8
チキンフィレオ	470	20.2
ベーコンレタスバーガー	374	18.1
チキンマックナゲット 5 ピース	263	15.3
えだまめコーン	83	5.2

0-1 ナップサック問題としての最適解

- **チーズバーガー，ビッグマック，フィレオフィッシュ，ベーコンレタスバーガー，チキンマックナゲット，えだまめコーン**
- タンパク質 **95.3 g**，カロリー 1893 kcal

無制限ナップサック問題としての最適解

- チーズバーガー 1 個，えだまめコーン 19 個
- タンパク質 114.5 g，カロリー 1884 kcal

ナップサック問題の例 (続き)

マクドナルドのメニューの栄養情報

種類	カロリー (kcal)	タンパク質 (g)
ハンバーガー	256	12.8
チーズバーガー	307	15.7
ビッグマック	530	26.4
フィレオフィッシュ	336	14.6
てりたまバーガー	566	21.8
チキンフィレオ	470	20.2
ベーコンレタスバーガー	374	18.1
チキンマックナゲット 5 ピース	263	15.3
えだまめコーン	83	5.2

0-1 ナップサック問題としての最適解

- チーズバーガー，ビッグマック，フィレオフィッシュ，ベーコンレタスバーガー，チキンマックナゲット，えだまめコーン
- タンパク質 95.3 g，カロリー 1893 kcal

無制限ナップサック問題としての最適解

- **チーズバーガー 1 個，えだまめコーン 19 個**
- タンパク質 **114.5 g**，カロリー 1884 kcal

組合せ最適化問題：最短路問題

距離がもっとも短い経路を求める問題

最短路問題 (shortest path problem)

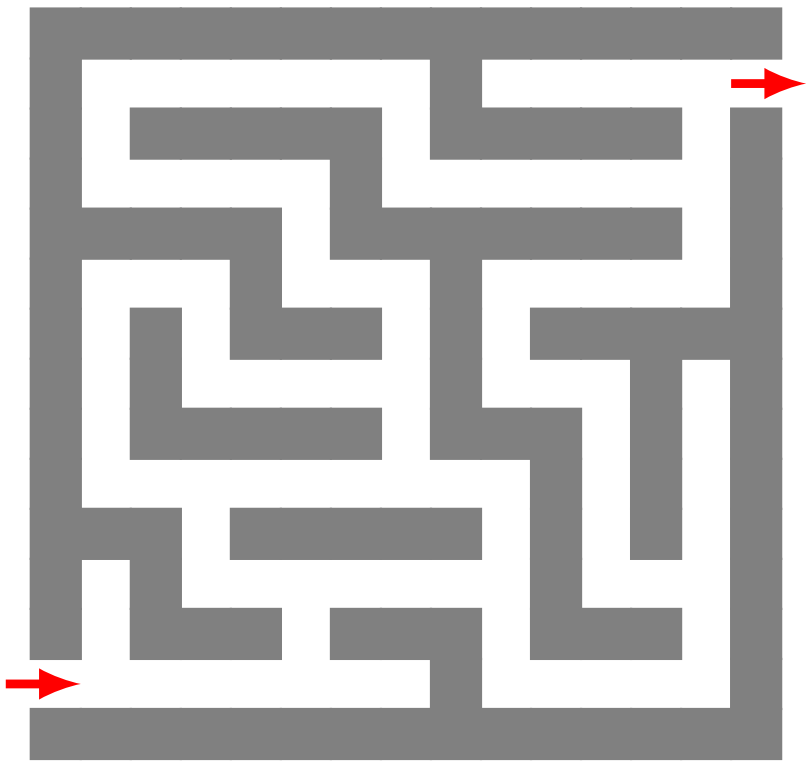
- n 個の地点
- 地点 i, j 間の距離：$d(i, j) \geq 0$
- 地点 1 から地点 n に至る最短路を求める

最短路問題の種類

- 単一点対 (single-pair) 最短路問題
  上の問題
- 単一始点 (single-source) 最短路問題
  ある地点から残りすべての地点への最短路を求める問題
- 全点対 (all-pairs) 最短路問題
  すべての 2 地点の組に対して最短路を求める問題

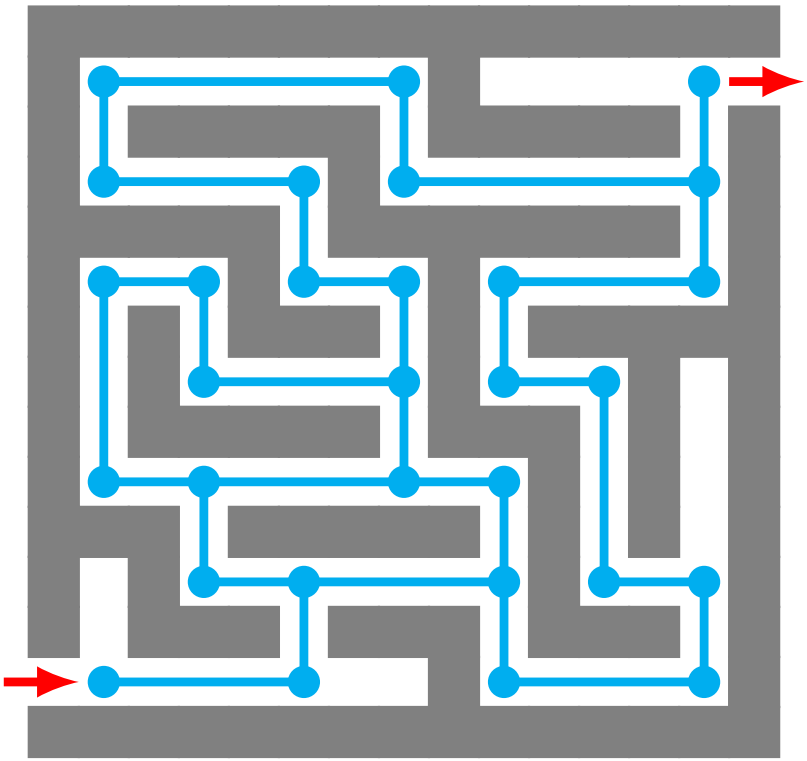
最短路問題の例

- スタートからゴールまで，最短距離でたどり着く
- 距離は移動マス数



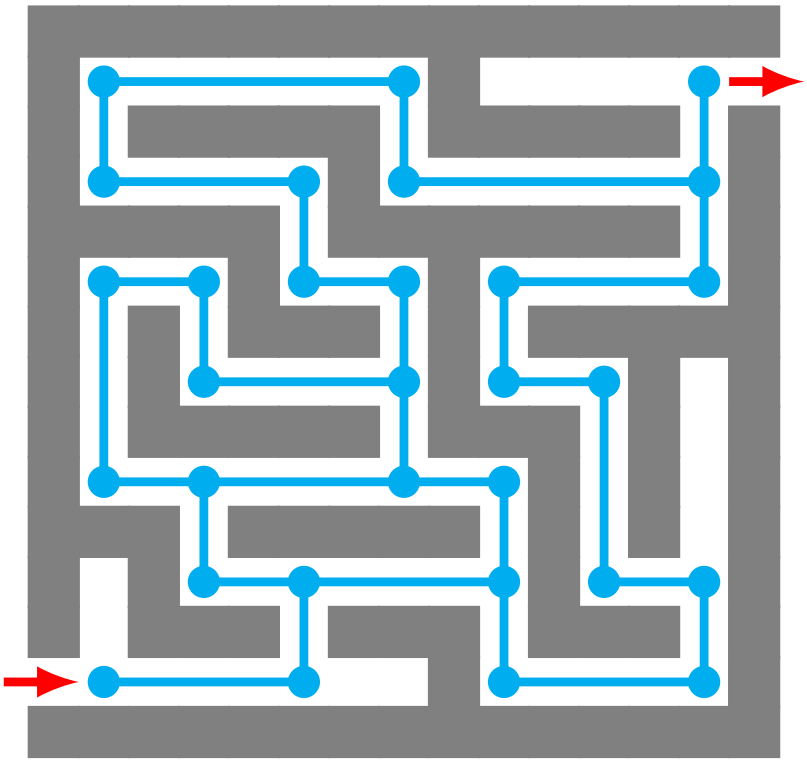
最短路問題の例

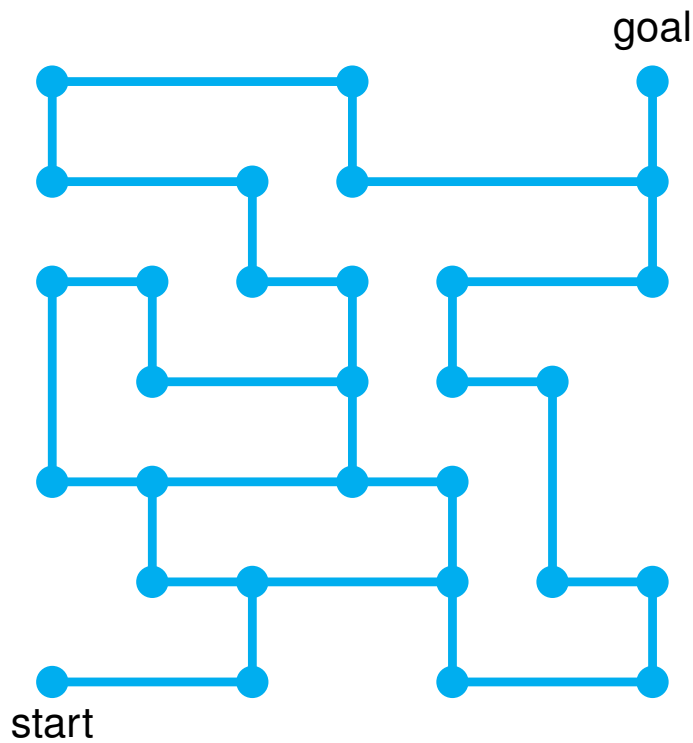
- スタートからゴールまで，最短距離でたどり着く
- 距離は移動マス数



最短路問題の例

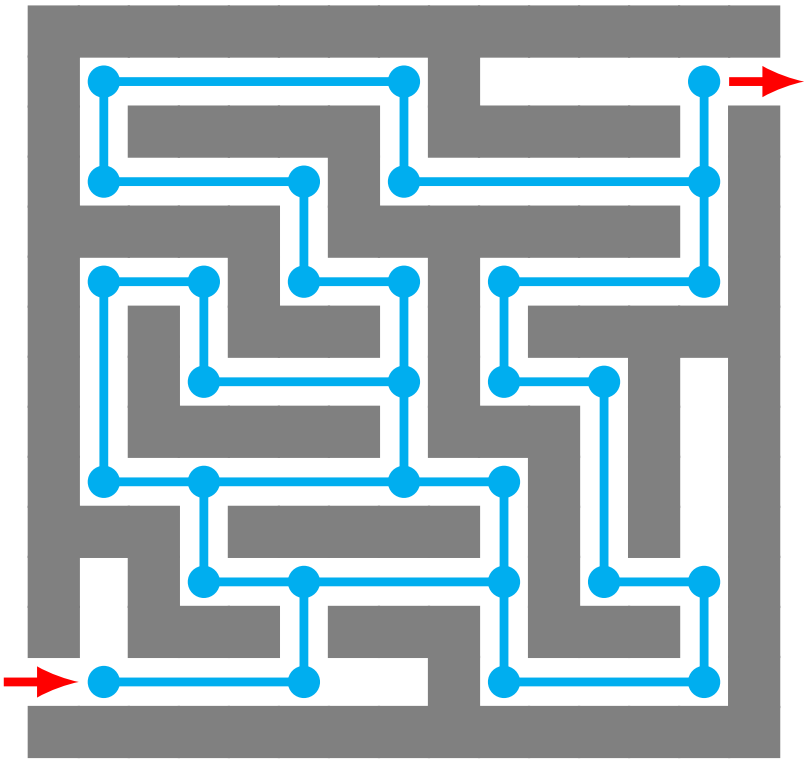
- スタートからゴールまで，最短距離でたどり着く
- 距離は移動マス数

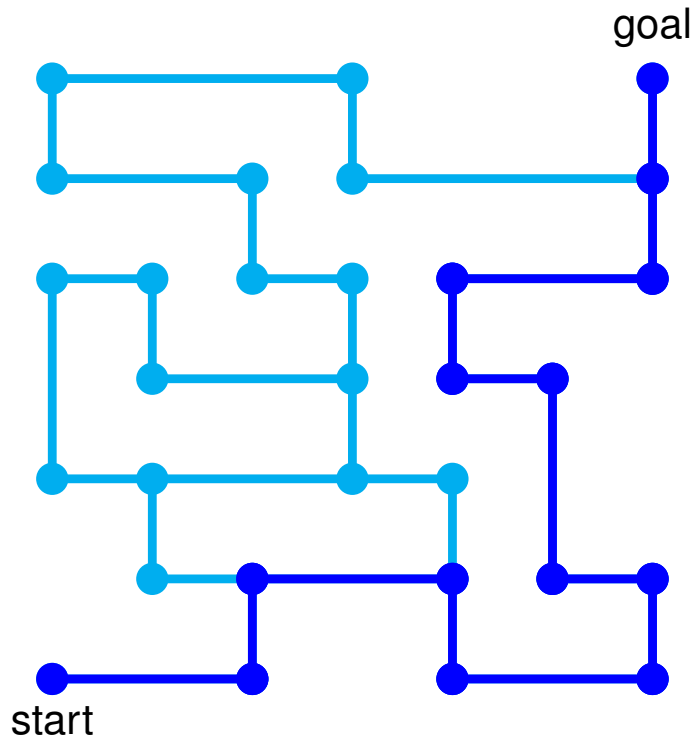




最短路問題の例

- スタートからゴールまで，最短距離でたどり着く
- 距離は移動マス数





最適解

36 マス移動

組合せ最適化問題：巡回セールスマン問題

巡回セールスマン問題 (traveling salesman problem)

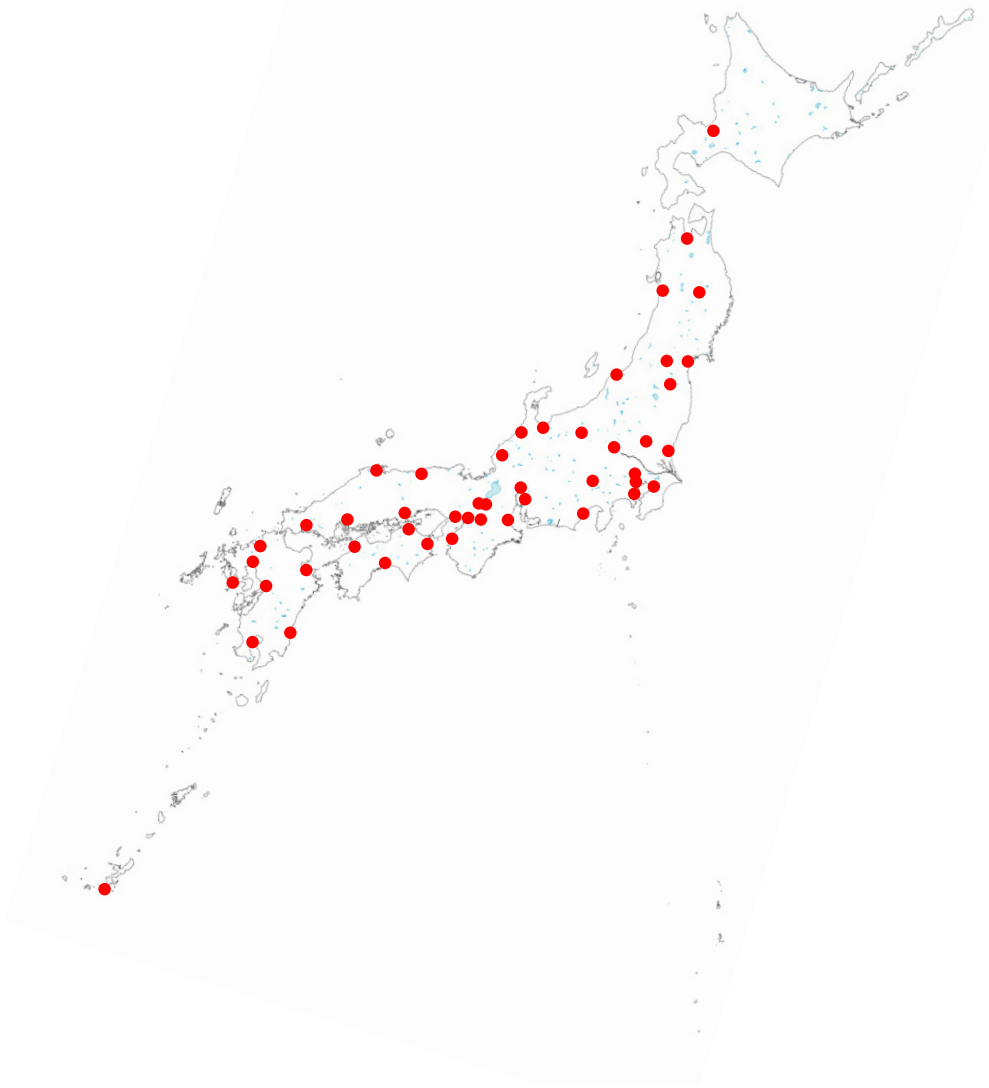
すべての地点を 1 回ずつ巡って出発地点に戻る経路 (巡回路) の中で，最短のものを求める問題

巡回セールスマン問題 (TSP) の種類

- 対称 TSP (symmetric TSP)
  地点 A→ 地点 B の距離と地点 B→ 地点 A の距離が同じ
- 非対称 TSP (asymmetric TSP)
  地点 A→ 地点 B の距離と地点 B→ 地点 A の距離が異なってもよい
- メトリック TSP (metric TSP)
  3 地点間の距離が三角不等式 (triangle inequality) を満たす
- ユークリッド TSP (Euclidean TSP)
  地点間の距離がユークリッド距離 (各地点の 2 次元座標で決まる)

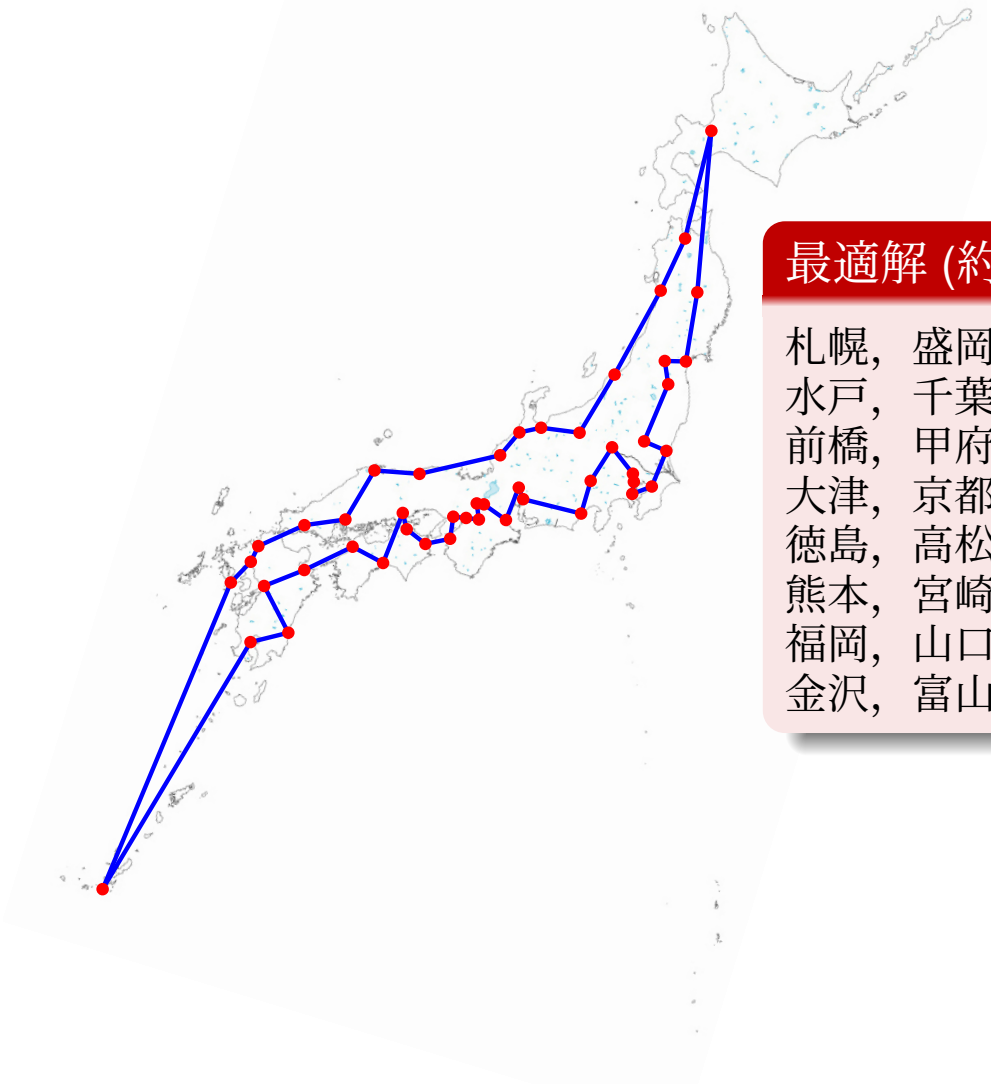
巡回セールスマン問題の例

- 県庁所在地の都市を 1 回ずつ巡って出発点に戻る最短巡回路を求める
- 都市間の距離は直線距離



巡回セールスマン問題の例

- 県庁所在地の都市を 1 回ずつ巡って出発点に戻る最短巡回路を求める
- 都市間の距離は直線距離

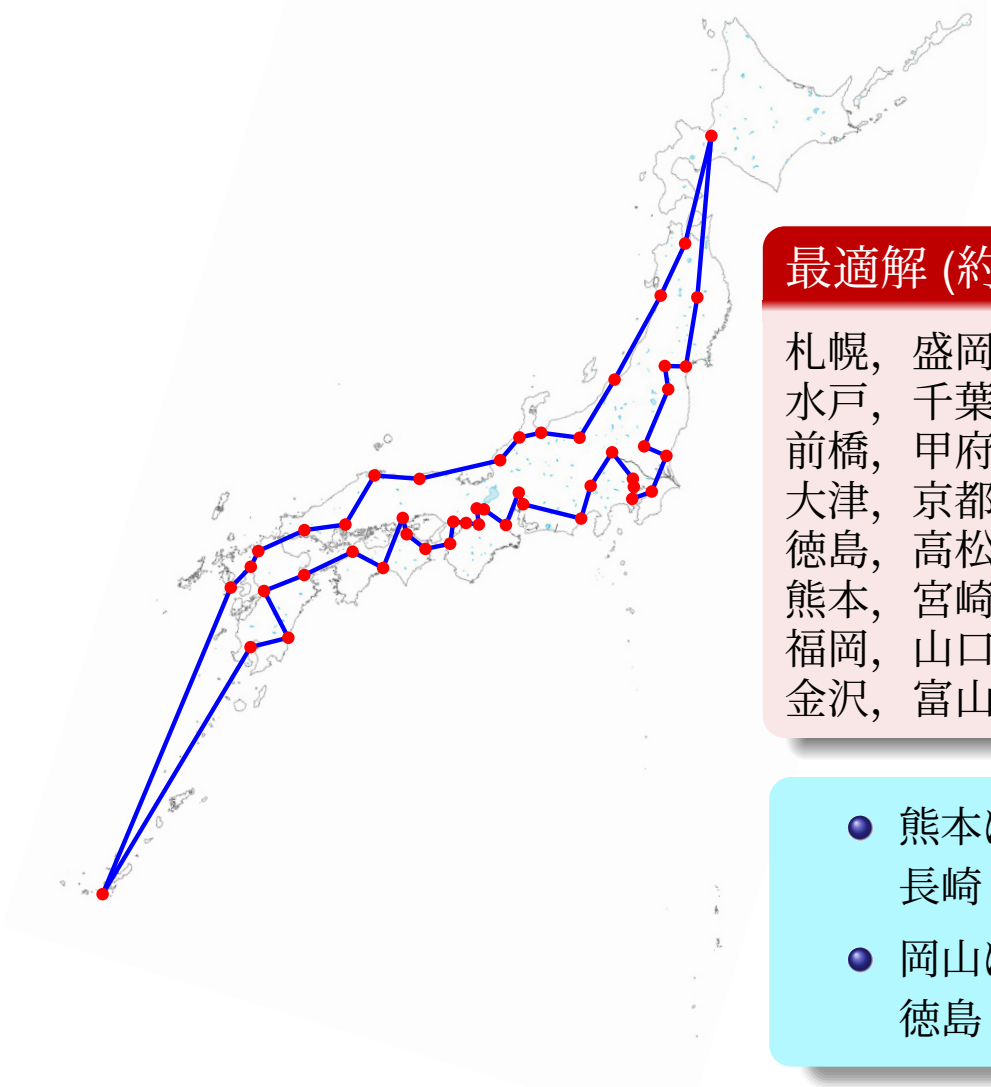


最適解 (約 5655 km)

札幌，盛岡，仙台，山形，福島，宇都宮，
水戸，千葉，横浜，新宿，さいたま，
前橋，甲府，静岡，名古屋，岐阜，津，
大津，京都，奈良，大阪，神戸，和歌山，
徳島，高松，岡山，高知，松山，大分，
熊本，宮崎，鹿児島，那覇，長崎，佐賀，
福岡，山口，広島，松江，鳥取，福井，
金沢，富山，長野，新潟，秋田，青森

巡回セールスマン問題の例

- 県庁所在地の都市を 1 回ずつ巡って出発点に戻る最短巡回路を求める
- 都市間の距離は直線距離



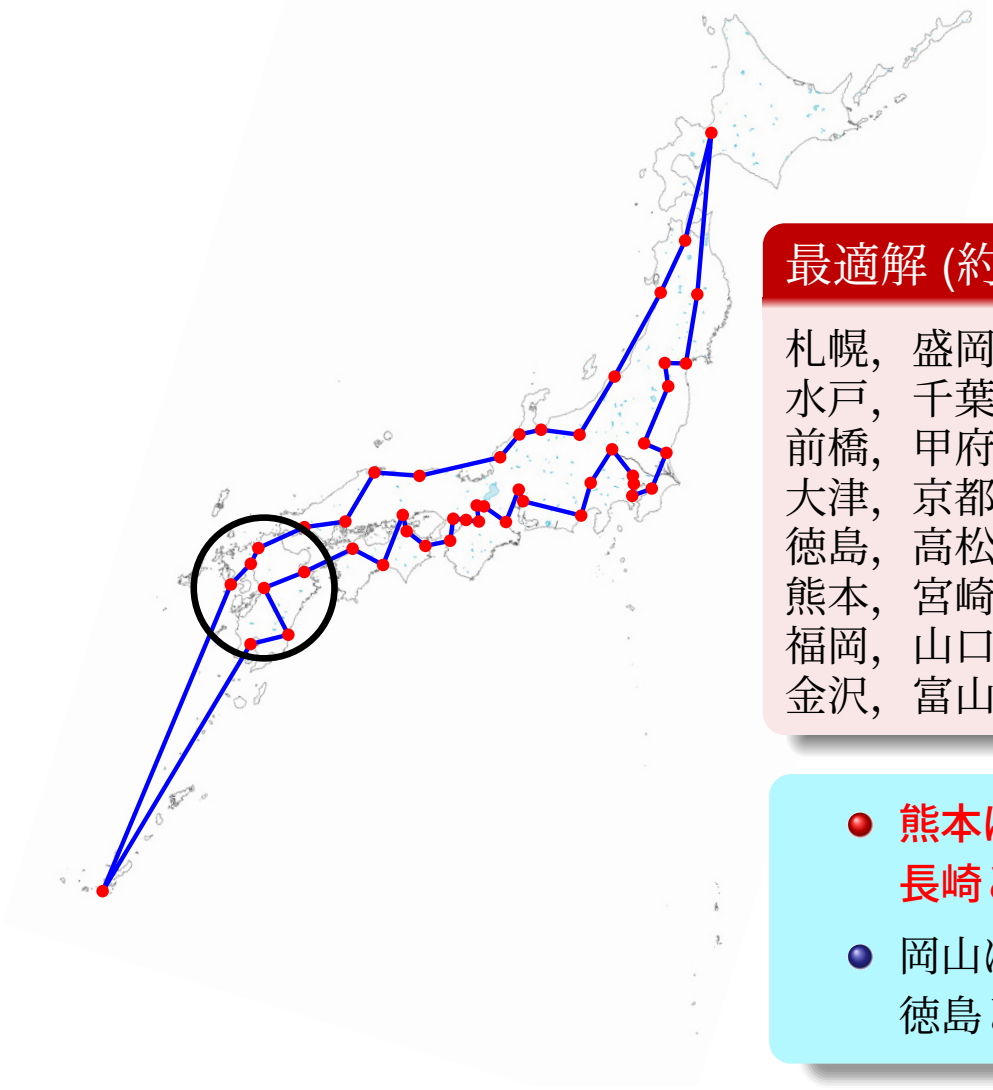
最適解 (約 5655 km)

札幌，盛岡，仙台，山形，福島，宇都宮，
水戸，千葉，横浜，新宿，さいたま，
前橋，甲府，静岡，名古屋，岐阜，津，
大津，京都，奈良，大阪，神戸，和歌山，
徳島，高松，岡山，高知，松山，大分，
熊本，宮崎，鹿児島，那覇，長崎，佐賀，
福岡，山口，広島，松江，鳥取，福井，
金沢，富山，長野，新潟，秋田，青森

- 熊本は大分と宮崎の間に訪れる方が得
  長崎と佐賀の間だと 5 km 増
- 岡山は高松と高知の間に訪れる方が得
  徳島と高松の間だと 3 km 増

巡回セールスマン問題の例

- 県庁所在地の都市を 1 回ずつ巡って出発点に戻る最短巡回路を求める
- 都市間の距離は直線距離



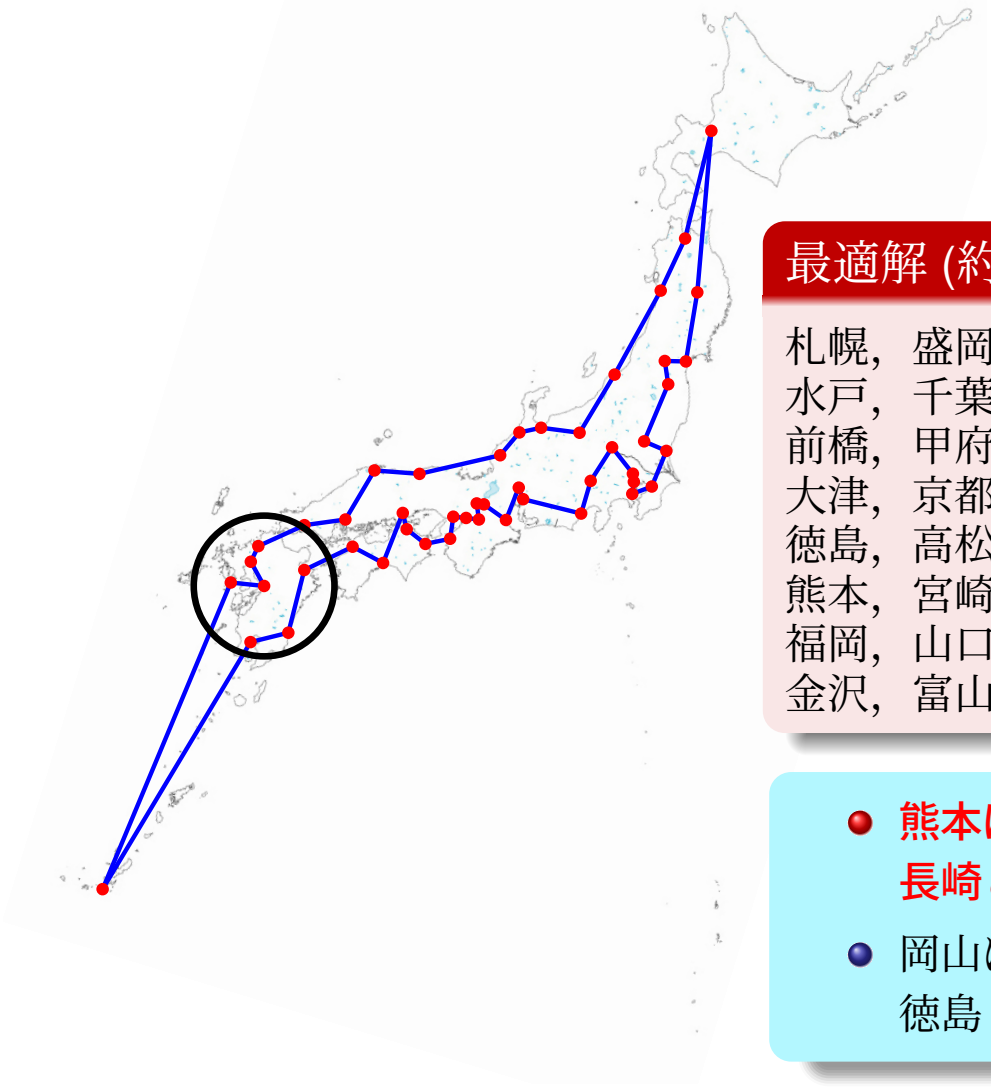
最適解 (約 5655 km)

札幌，盛岡，仙台，山形，福島，宇都宮，
水戸，千葉，横浜，新宿，さいたま，
前橋，甲府，静岡，名古屋，岐阜，津，
大津，京都，奈良，大阪，神戸，和歌山，
徳島，高松，岡山，高知，松山，大分，
熊本，宮崎，鹿児島，那覇，長崎，佐賀，
福岡，山口，広島，松江，鳥取，福井，
金沢，富山，長野，新潟，秋田，青森

- **熊本は大分と宮崎の間に訪れる方が得**
  長崎と佐賀の間だと 5 km 増
- 岡山は高松と高知の間に訪れる方が得
  徳島と高松の間だと 3 km 増

巡回セールスマン問題の例

- 県庁所在地の都市を 1 回ずつ巡って出発点に戻る最短巡回路を求める
- 都市間の距離は直線距離



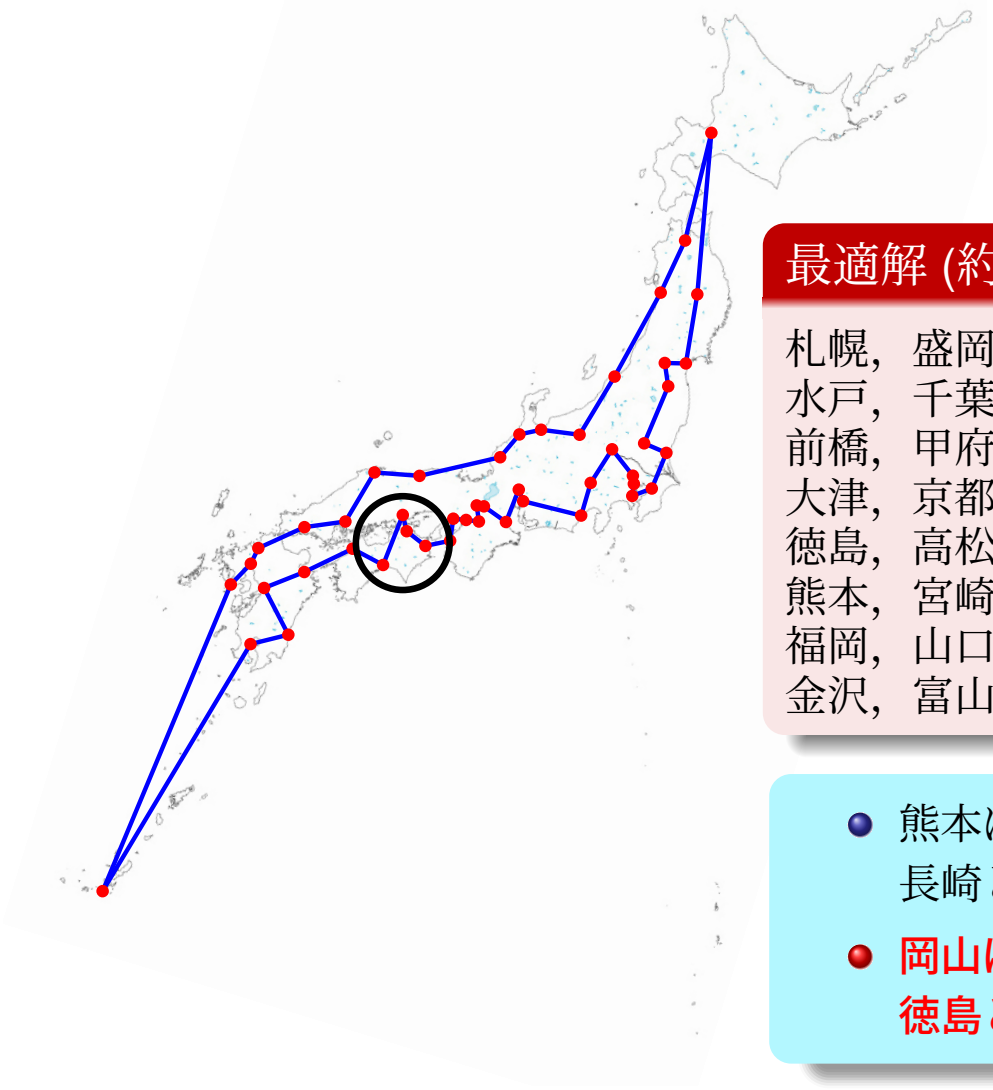
最適解 (約 5655 km)

札幌，盛岡，仙台，山形，福島，宇都宮，
水戸，千葉，横浜，新宿，さいたま，
前橋，甲府，静岡，名古屋，岐阜，津，
大津，京都，奈良，大阪，神戸，和歌山，
徳島，高松，岡山，高知，松山，大分，
熊本，宮崎，鹿児島，那覇，長崎，佐賀，
福岡，山口，広島，松江，鳥取，福井，
金沢，富山，長野，新潟，秋田，青森

- **熊本は大分と宮崎の間に訪れる方が得**
  長崎と佐賀の間だと 5 km 増
- 岡山は高松と高知の間に訪れる方が得
  徳島と高松の間だと 3 km 増

巡回セールスマン問題の例

- 県庁所在地の都市を 1 回ずつ巡って出発点に戻る最短巡回路を求める
- 都市間の距離は直線距離



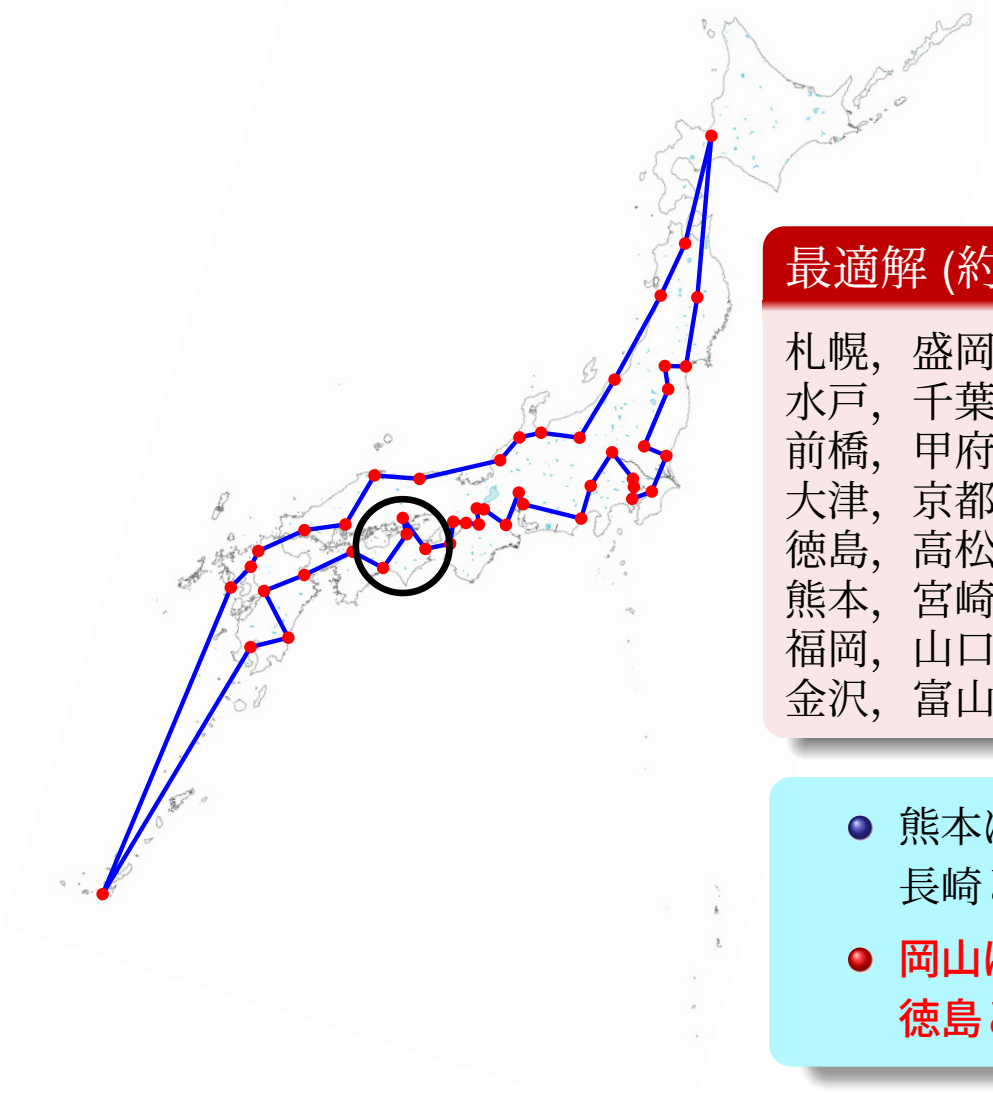
最適解 (約 5655 km)

札幌，盛岡，仙台，山形，福島，宇都宮，
水戸，千葉，横浜，新宿，さいたま，
前橋，甲府，静岡，名古屋，岐阜，津，
大津，京都，奈良，大阪，神戸，和歌山，
徳島，高松，岡山，高知，松山，大分，
熊本，宮崎，鹿児島，那覇，長崎，佐賀，
福岡，山口，広島，松江，鳥取，福井，
金沢，富山，長野，新潟，秋田，青森

- 熊本は大分と宮崎の間に訪れる方が得
  長崎と佐賀の間だと 5 km 増
- **岡山は高松と高知の間に訪れる方が得**
  徳島と高松の間だと 3 km 増

巡回セールスマン問題の例

- 県庁所在地の都市を 1 回ずつ巡って出発点に戻る最短巡回路を求める
- 都市間の距離は直線距離



最適解 (約 5655 km)

札幌，盛岡，仙台，山形，福島，宇都宮，
水戸，千葉，横浜，新宿，さいたま，
前橋，甲府，静岡，名古屋，岐阜，津，
大津，京都，奈良，大阪，神戸，和歌山，
徳島，高松，岡山，高知，松山，大分，
熊本，宮崎，鹿児島，那覇，長崎，佐賀，
福岡，山口，広島，松江，鳥取，福井，
金沢，富山，長野，新潟，秋田，青森

- 熊本は大分と宮崎の間に訪れる方が得
  長崎と佐賀の間だと 5 km 増
- **岡山は高松と高知の間に訪れる方が得**
  徳島と高松の間だと 3 km 増

組合せ最適化問題：ハミルトン閉路問題

ハミルトン閉路 (Hamiltonian cycle, Hamiltonian circuit)

- すべての地点をちょうど 1 回ずつ巡って出発地点に戻る巡回路
- ハミルトン閉路問題 (Hamiltonian cycle problem)
  - ハミルトン閉路が存在するかどうかを判定する問題
  - 直接接続している地点の組は一部のみ
    例：北海道-沖縄間は直接行き来できない
- 巡回セールスマン問題の解はハミルトン閉路

オイラー路・オイラー閉路

- オイラー路 (Eulerian trail, Eulerian path)
  すべての道路をちょうど 1 回ずつ通る経路
- オイラー閉路 (Eulerian cycle, Eulerian circuit)
  すべての道路をちょうど 1 回ずつ通って出発地点に戻る巡回路

中国人郵便配達問題 (Chinese postman problem)

- すべての道路を**少なくとも 1 回ずつ**通って出発地点に戻る最短の巡回路
- 中国人郵便配達問題の解はオイラー閉路に類似 (少し違う)

グラフ

地点間の接続関係を考える問題はグラフを用いて表すのが便利

- 最短経路問題
- 巡回セールスマン問題
- ハミルトン閉路問題
- etc.

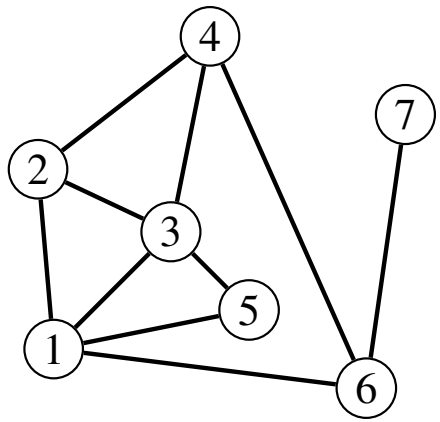
グラフ (graph)

- グラフ $G = (V, E)$
- V: 頂点 (vertex)，節点 (node) の集合
  - 地点 $\Rightarrow$ 頂点
- E: 辺，枝 (edge) の集合
  - 頂点 (地点) 間の接続関係を表す
  - 頂点 u と頂点 v が直接接続 $\Rightarrow$ 辺 (u, v)

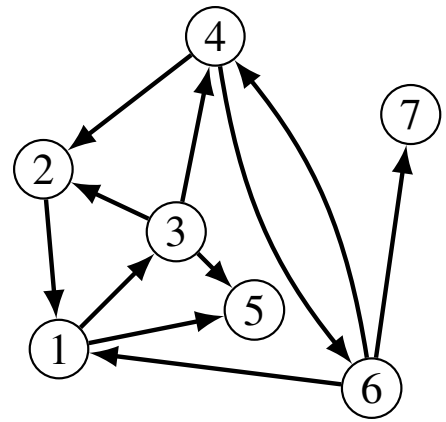
無向グラフと有向グラフ

無向グラフと有向グラフ

- **無向グラフ** (undirected graph)
  - 辺に方向がない
  - (u, v) と (v, u) は同じ辺
- **有向グラフ** (directed graph, digraph)
  - (u, v) は $u \to v$ の向き，(v, u) は $v \to u$ の向き
  - 有向グラフの辺は有向辺や孤 (arc) とも



無向グラフ



有向グラフ

$G = (V, E)$, $V = \{1, 2, 3, 4, 5, 6, 7\}$,

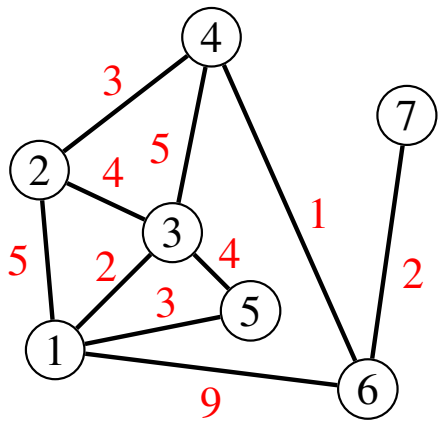
無向グラフ $E = \{(1, 2), (1, 3), (1, 5), (1, 6), (2, 3), (2, 4), (3, 4), (3, 5), (4, 6), (6, 7)\}$

有向グラフ $E = \{(1, 3), (1, 5), (2, 1), (3, 2), (3, 4), (3, 5), (4, 2), (4, 6), (6, 1), (6, 4), (6, 7)\}$

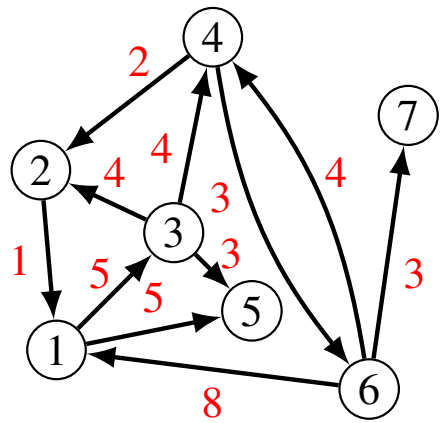
ネットワーク

ネットワーク (network)

辺に重み (距離やコストなど) を付加したグラフ

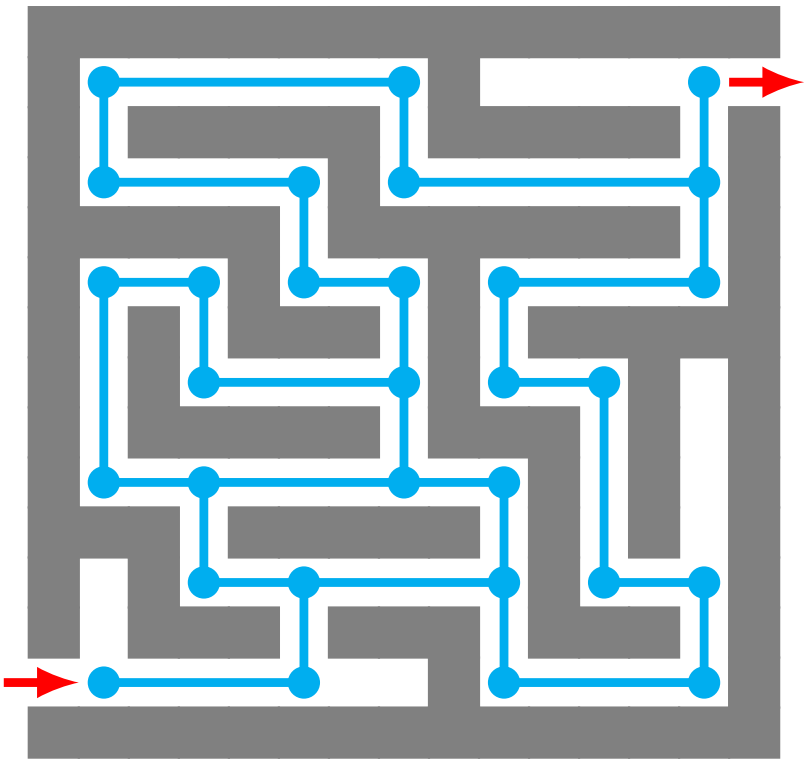


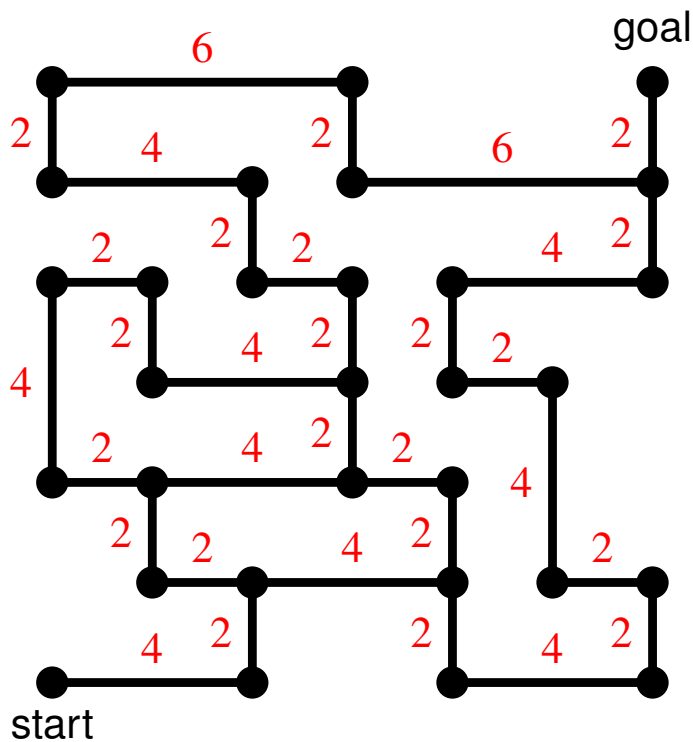
無向グラフ



有向グラフ

ネットワークの例：最短経路問題





計算複雑さと計算量

計算複雑さと計算量

- 組合せ最適化問題の難易度は様々
  - 難しい問題
  - 簡単な問題
- 解き方 (アルゴリズム) も様々

計算量

- アルゴリズムの計算効率の評価指標
  - 計算時間
  - 記憶領域の使用量
- 漸近表記：$O(n^2)$ や $\Theta(n \log n)$ など
  - 最高次の項のみ
  - 低次元の項や定数は無視

問題のクラス

- 解きやすさ・解きにくさで問題を分類
- 使うアルゴリズムに依存するが，工夫しても解きにくい問題が存在

整数計画問題 (integer programming problem)

整数計画問題 (integer programming problem)

- 決定変数が整数値のみを取る最適化問題
- 組合せ最適化問題の多くは整数計画問題として表現可能

整数計画問題の分類

- 混合整数計画問題 (mixed-integer programming problem; MIP)
  連続値と整数値の決定変数が混在
- **整数線形計画問題** (integer linear programming problem; ILP)
  目的関数が線形，制約条件が 1 次式
- **混合整数線形計画問題** (mixed-integer linear programming problem; MILP)
  混合整数計画問題かつ整数線形計画問題

(混合) 整数線形計画問題の解法

- 分枝限定法
- 切除平面法

いずれも，整数制約を取り除く (緩和する) と線形計画問題となることを利用

整数計画問題：割当問題

- アルバイト 1, 2, 3 を，シフト 1, 2, 3 に一人ずつ割り当てる
- 総コスト最小の勤務表を作成

割り当てコスト

バイト \ シフト	1	2	3
1	c_{11}	c_{12}	c_{13}
2	c_{21}	c_{22}	c_{23}
3	c_{31}	c_{32}	c_{33}

決定変数 x_{ij}: バイト i のシフト j への割り当て

$$x_{ij} = \begin{cases} 1, & \text{バイト } i \text{ をシフト } j \text{ に割り当てるとき} \\ 0, & \text{そうでないとき} \end{cases}$$

$$\begin{aligned} \min \quad & \sum_{i=1}^{3} \sum_{j=1}^{3} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{3} x_{ij} = 1, \qquad 1 \le i \le 3 \\ & \sum_{i=1}^{3} x_{ij} = 1, \qquad 1 \le j \le 3 \\ & x_{ij} \in \{0, 1\}, \qquad 1 \le i \le 3,\ 1 \le j \le 3 \end{aligned}$$

整数計画問題：割当問題

- アルバイト 1, 2, 3 を，シフト 1, 2, 3 に一人ずつ割り当てる
- 総コスト最小の勤務表を作成

割り当てコスト

バイト \ シフト	1	2	3
1	c_{11}	c_{12}	c_{13}
2	c_{21}	c_{22}	c_{23}
3	c_{31}	c_{32}	c_{33}

決定変数 x_{ij}: バイト i のシフト j への割り当て

$$x_{ij} = \begin{cases} 1, & \text{バイト } i \text{ をシフト } j \text{ に割り当てるとき} \\ 0, & \text{そうでないとき} \end{cases}$$

$$\min \quad \sum_{i=1}^{3} \sum_{j=1}^{3} c_{ij} x_{ij} \qquad \text{(総コスト)}$$

$$\text{s.t.} \quad \sum_{j=1}^{3} x_{ij} = 1, \quad 1 \le i \le 3 \qquad \text{(バイト } i \text{ のシフトはちょうど一つ)}$$

$$\sum_{i=1}^{3} x_{ij} = 1, \quad 1 \le j \le 3 \qquad \text{(シフト } j \text{ のバイトはちょうど一人)}$$

$$x_{ij} \in \{0, 1\}, \quad 1 \le i \le 3,\ 1 \le j \le 3 \qquad \text{(決定変数の範囲)}$$

練習問題：0-1 ナップサック問題

- 4 つのアイテム 1, 2, 3, 4，容量 C のナップサック
- アイテム i のサイズ a_i，利得 c_i
- 容量を超えないようアイテムをナップサックに詰め込んで，総利得を最大化
- 同じアイテムを重複して選択することはできない

練習問題：0-1 ナップサック問題

- 4 つのアイテム 1, 2, 3, 4，容量 C のナップサック
- アイテム i のサイズ a_i，利得 c_i
- 容量を超えないようアイテムをナップサックに詰め込んで，総利得を最大化
- 同じアイテムを重複して選択することはできない

決定変数 x_i: アイテム i の選択・非選択

$$x_i = \begin{cases} 1, & \text{アイテム } i \text{ をナップサックに詰めるとき} \\ 0, & \text{そうでないとき} \end{cases}$$

練習問題：0-1 ナップサック問題

- 4 つのアイテム 1, 2, 3, 4，容量 C のナップサック
- アイテム i のサイズ a_i，利得 c_i
- 容量を超えないようアイテムをナップサックに詰め込んで，総利得を最大化
- 同じアイテムを重複して選択することはできない

決定変数 x_i: アイテム i の選択・非選択

$$x_i = \begin{cases} 1, & \text{アイテム } i \text{ をナップサックに詰めるとき} \\ 0, & \text{そうでないとき} \end{cases}$$

$$\max \quad \sum_{i=1}^{4} c_i x_i \qquad \text{(総利得)}$$

$$\text{s.t.} \quad \sum_{i=1}^{4} a_i x_i \le C, \qquad \text{(サイズの合計が容量を超えない)}$$

$$x_i \in \{0, 1\}, \qquad 1 \le i \le 4 \qquad \text{(決定変数の範囲)}$$