大阪電気通信大学 情報通信工学部 光システム工学科 2年次配当科目

コンピュータアルゴリズム

グラフ (2)

第10講: 平成20年12月19日 (金) 4限 E252教室

中村 嘉隆(なかむら よしたか)
奈良先端科学技術大学院大学 助教
y-nakamr@is.naist.jp
http://narayama.naist.jp/~y-nakamr/

第 11 講の復習

- グラフ(graph)
  - グラフとは
  - グラフの表現
  - グラフの探索
    - 深さ優先探索
    - 幅優先探索

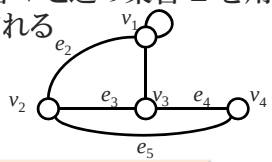
復習: グラフとは

- グラフ: Graph
- 頂点(vertex, node, 節点)の集合と辺(edge, arc, branch, 枝)の集合からなる
- グラフ G は頂点の集合 V と辺の集合 E を用いて, $G = (V, E)$ と表される

$G = (V, E)$

$V = \{v_1, v_2, v_3, v_4\}$

$E = \{e_1, e_2, e_3, e_4, e_5\}$



頂点 v_i, v_j 間の辺 e_{ij} を 辺 (v_i, v_j) と書く

復習: グラフの種類

- 有向グラフ
  - 辺に向きのあるグラフ
- 無向グラフ
  - 辺に向きのないグラフ

有向グラフの場合, 辺 (v_1, v_2) と辺 (v_2, v_1) は別物

- 重みつきグラフ
  - 辺の属性として重み(コスト, 長さ)を持つグラフ
- 道(path, 路)
  - 頂点と頂点を結ぶ経路
- 閉路(cycle, closed path)
  - 同じ頂点へ帰ってくる道
  - 自分自身の頂点への辺は自己閉路(self loop)という

復習: グラフの用語

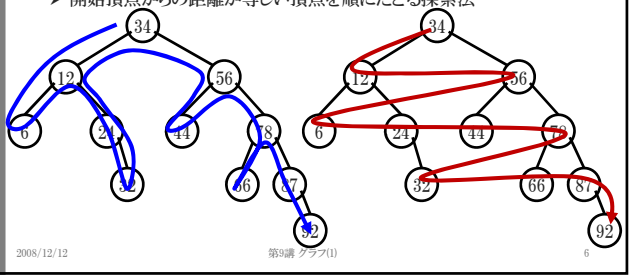
- 次数
  - 頂点に接続されている辺の数
  - 有向グラフの場合は, 入ってくる辺と出て行く辺を区別して, それぞれ入次数, 出次数という
- 完全グラフ
  - すべての頂点間に辺があるグラフ
- 隣接行列
  - 各頂点間に辺があれば 1, なければ 0 とした行列
- 探索
  - グラフ上のすべての頂点を訪問すること

次数 5
(入次数 2, 出次数 3)

	v_1	v_2	v_3	v_4
v_1	1	1	1	0
v_2	0	1	0	1
v_3	0	1	1	1
v_4	0	0	0	1

復習: 探索アルゴリズム

- 深さ優先探索(左図)
  - 開始頂点から, 一つの道を選んでいけるところまで行き, 進めなくなったら引き返して別の道を選ぶ探索法
- 幅優先探索(右図)
  - 開始頂点からの距離が等しい頂点を順にたどる探索法



本日の講義内容

- グラフアルゴリズムの紹介
  - 最短路の問題
    - ダイクストラのアルゴリズム
  - 最小木の問題
    - プリムのアルゴリズム

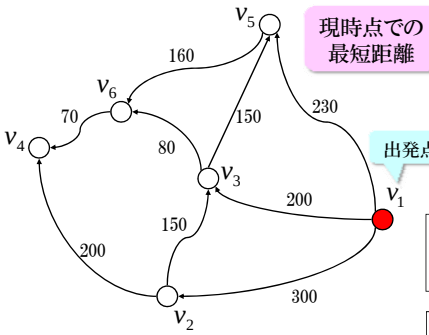
最短路を求める問題

- ダイクストラ(Dijkstra)のアルゴリズム
  - ある頂点 s から他の各頂点への最短経路を求めるための効率の良いアルゴリズム
  - SPF(Shortest Path First)アルゴリズムとも呼ばれる
  - インターネットのルータ間の経路制御や，駅すぱーとなどの乗換案内ソフトウェア，カーナビ，ゲーム(桃太郎電鉄)などでも使われている
  - エドガー・ダイクストラ(Edsger Wybe Dijkstra)
    - オランダ人情報工学者，1930年-2002年
    - 1972年，プログラミング言語の基礎研究に対してチューリング賞を受賞．構造化プログラミングの提唱者．

ダイクストラのアルゴリズム

- 概要
  - 重み(ただし正の数)のつけられたグラフにおいて最短経路を求めるアルゴリズム
  - 対象はすべての頂点が連結されたグラフとする
- 基本戦略
  - 各頂点の最短経路を出発点に近い(最短経路の長さが短い)ものから一つずつ確定していく
- 性質
  - 最短経路が(経路があれば)必ず見つかることが保証されている

ダイクストラ法 － 初期状態 －



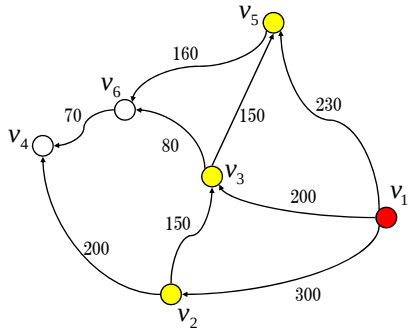
f1	0	－
f2	∞	－
f3	∞	－
f4	∞	－
f5	∞	－
f6	∞	－

P:最短距離計算済頂点

T:次候補頂点

v_1

ダイクストラ法 － ステップ 1 －



f1	0	－
f2	300	1
f3	200	1
f4	∞	－
f5	230	1
f6	∞	－

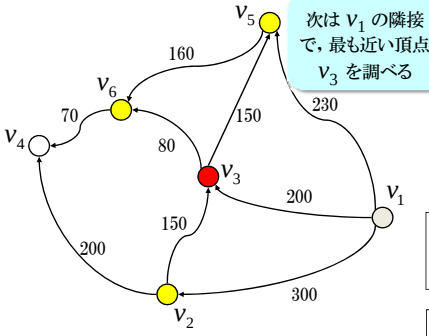
P:最短距離計算済頂点

v_1

T:次候補頂点

v_2 v_3 v_5

ダイクストラ法 － ステップ 2 －



f1	0	－
f2	300	1
f3	200	1
f4	∞	－
f5	230	1
f6	280	3

P:最短距離計算済頂点

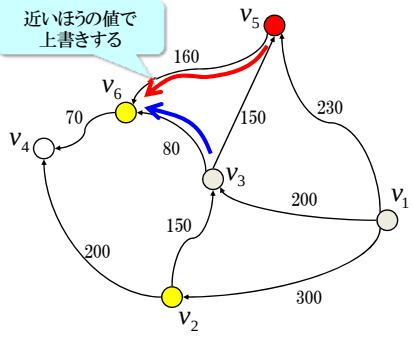
v_1 v_3

T:次候補頂点

v_2 v_5 v_6

ダイクストラ法 － ステップ 3 －

近いほうの値で上書きする



f1	0	－
f2	300	1
f3	200	1
f4	∞	－
f5	230	1
f6	280	3

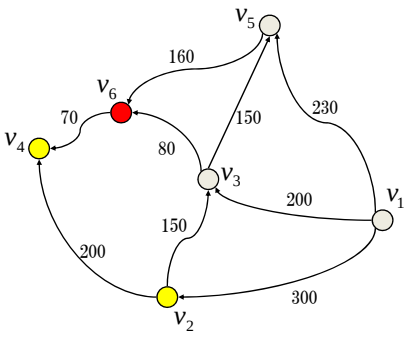
P:最短距離計算済頂点

v_1 v_3 v_5

T:次候補頂点

v_2 v_6

ダイクストラ法 － ステップ 4 －



f1	0	－
f2	300	1
f3	200	1
f4	350	6
f5	230	1
f6	280	3

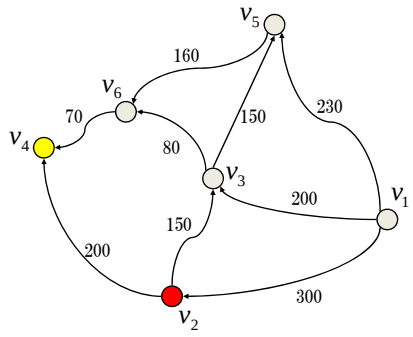
P:最短距離計算済頂点

v_1 v_3 v_5 v_6

T:次候補頂点

v_2 v_4

ダイクストラ法 － ステップ 5 －



f1	0	－
f2	300	1
f3	200	1
f4	350	6
f5	230	1
f6	280	3

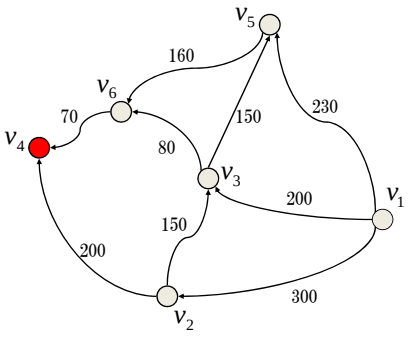
P:最短距離計算済頂点

v_1 v_3 v_5 v_6 v_2

T:次候補頂点

v_4

ダイクストラ法 － 終了 －



f1	0	－
f2	300	1
f3	200	1
f4	350	6
f5	230	1
f6	280	3

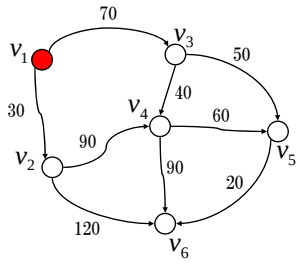
P:最短距離計算済頂点

v_1 v_3 v_5 v_6 v_2 v_4

T:次候補頂点

演習：ダイクストラ法

- 最短経路の表を完成させよ



f1	0	－
f2	∞	－
f3	∞	－
f4	∞	－
f5	∞	－
f6	∞	－

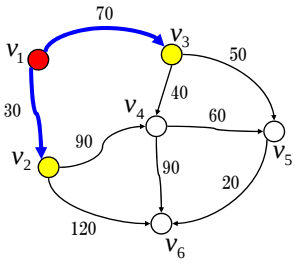
P:最短距離計算済頂点

T:次候補頂点

v_1

演習：ダイクストラ法 －ステップ 1－

- 最短経路の表を完成させよ



f1	0	－
f2	30	1
f3	70	1
f4	∞	－
f5	∞	－
f6	∞	－

P:最短距離計算済頂点

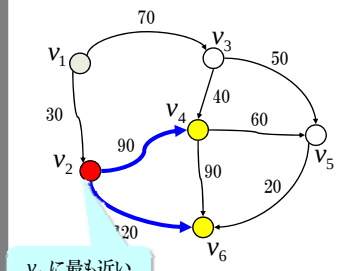
v_1

T:次候補頂点

v_2 v_3

演習：ダイクストラ法　-ステップ 2-

- 最短経路の表を完成させよ



f1	0	-
f2	30	1
f3	70	1
f4	120	2
f5	∞	-
f6	150	2

30+90

30+120

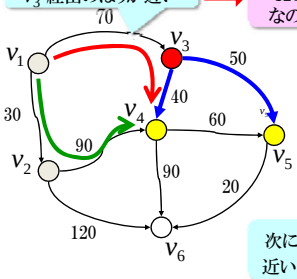
v_1 に最も近い v_2 を次は基準

P:最短距離計算済頂点: v_1 v_2

T:次候補頂点: v_3 v_4 v_6

演習：ダイクストラ法　-ステップ 3-

- 最短経路の表を完成させよ



f1	0	-
f2	30	1
f3	70	1
f4	110	3
f5	120	3
f6	150	2

v_3 経由のほうが近い

120 > 70+40 なので上書き

70+50

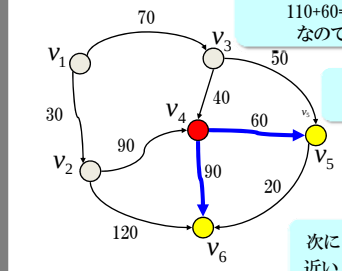
次に v_1 に最も近い v_3 を基準

P:最短距離計算済頂点: v_1 v_2 v_3

T:次候補頂点: v_4 v_6 v_5

演習：ダイクストラ法　-ステップ 4-

- 最短経路の表を完成させよ



f1	0	-
f2	30	1
f3	70	1
f4	110	3
f5	120	3
f6	150	2

v_4 を経由して v_5 に行くと 110+60=170 > 120 なのでそのまま

v_6 も同様 110+90 > 150

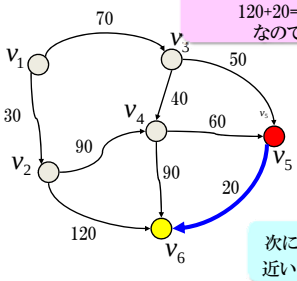
次に v_1 に最も近い v_4 を基準

P:最短距離計算済頂点: v_1 v_2 v_3 v_4

T:次候補頂点: v_6 v_5

演習：ダイクストラ法　-ステップ 5-

- 最短経路の表を完成させよ



f1	0	-
f2	30	1
f3	70	1
f4	110	3
f5	120	3
f6	140	5

v_5 を経由して v_6 に行くと 120+20=140 < 150 なので上書き

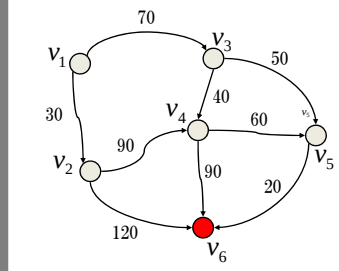
次に v_1 に最も近い v_5 を基準

P:最短距離計算済頂点: v_1 v_2 v_3 v_4 v_5

T:次候補頂点: v_6

演習：ダイクストラ法　-終了-

- 最短経路の表を完成させよ



f1	0	-
f2	30	1
f3	70	1
f4	110	3
f5	120	3
f6	140	5

P:最短距離計算済頂点: v_1 v_2 v_3 v_4 v_5 v_6

T:次候補頂点:

最小木を求める問題

- 最小木（MST: Minimum Spanning Tree）
  - すべての頂点を連結する木で，辺の重みの総和が最小のもの
  - 連結
    - すべての頂点間に経路が存在する
  - 木なので閉路は存在しない
- 応用例
  - ネットワークの接続
    - ケーブル長最小ですべてのコンピュータを接続する
  - 道路の敷設
    - 道路長最小（経路の平均最小ではない）で各戸を結ぶ

プリム(Prim)のアルゴリズム

- ダイクストラ法と基本は同じ
- 基本戦略
  - 最も重みの小さな辺から順に最小木の枝になるかどうか調べていく

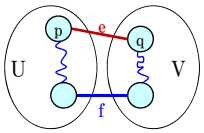
プリム(Prim)のアルゴリズム

```
Prim()
{
    V ← 空集合;
    U ← すべての頂点からなる集合;
    vertex[出発点].distance ← 0;
    for (x ← 出発点以外のすべての頂点について) {
        vertex[x].distance ← ∞;
    }
    while (U が空集合でない) {
        p ← U の中でフィールド distance が最小の頂点;
        p を U から除き, V に加える;
        for (p を始点とする全ての辺について) {
            x ← その辺の終点の頂点;
            if (x が U に属している) {
                vertex[x].distance ← Min(vertex[x].distance,
    その辺の重み);
            }
        }
    }
}
```

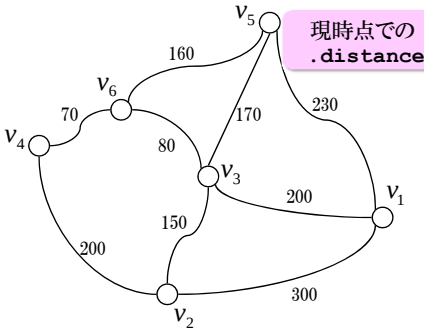
ちなみにここを，"vertex[p].distance + その辺の重み"
とするとダイクストラのアルゴリズムと同じになる

プリムのアルゴリズムの正当性

- プリム法では，各ステップで集合 V と U を結ぶ辺のうち，重み最小のものを選んでいる
- この辺を e とすると，これを最小木は必ず含む
- 背理法で証明
  - e を含まない最小木があったと仮定
  - 最小木は頂点全てを連結させるから，V と U を結ぶ辺が他に必ず存在
  - それを f とすると，f の重みは e 以上
  - f を含む最小木には e の端点となる頂点 p, q を結ぶ経路が存在
  - e をその最小木に加えると，閉路ができてしまう
  - そこから f を除けば閉路がなくなり，また木が得られる
  - この木は f が入っているときより，重みの総和が小さくなる
  - これは仮定に矛盾
- プラム法はこの事実を基に構成されている



プリム法 - 初期状態



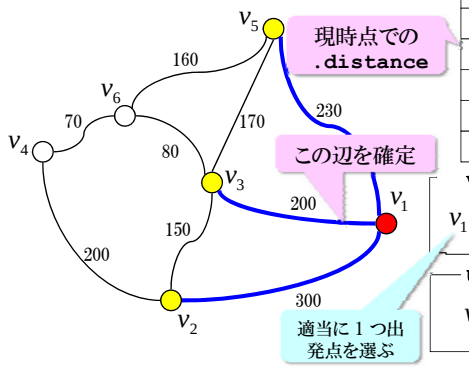
f1	∞
f2	∞
f3	∞
f4	∞
f5	∞
f6	∞

V:調査済頂点

U:未調査頂点

v_1 v_2 v_3 v_4 v_5 v_6

プリム法 - ステップ 1



f1	0
f2	300
f3	200
f4	∞
f5	230
f6	∞

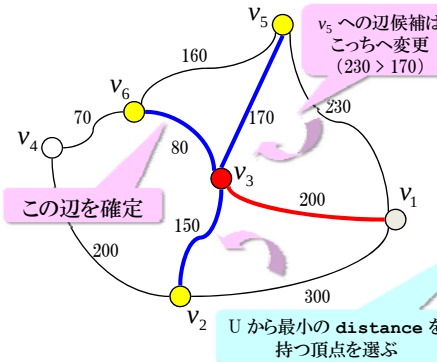
V:調査済頂点

v_1

U:未調査頂点

v_2 v_3 v_4 v_5 v_6

プリム法 - ステップ 2



f1	0
f2	150
f3	200
f4	∞
f5	170
f6	80

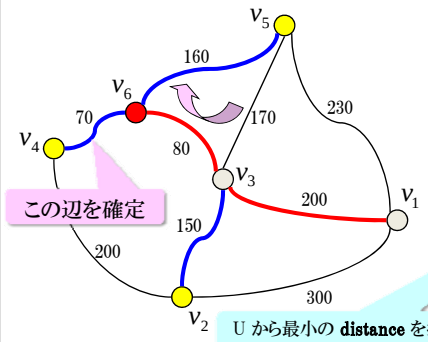
V:調査済頂点

v_1 v_3

U:未調査頂点

v_2 v_4 v_5 v_6

プリム法 - ステップ 3 -



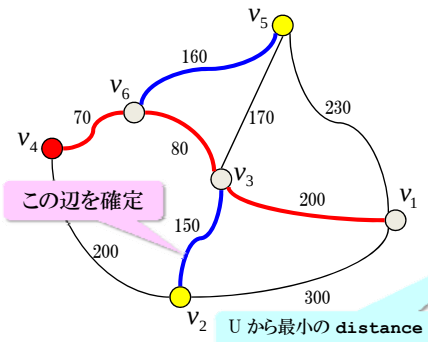
f1	0
f2	150
f3	200
f4	70
f5	160
f6	80

V:調査済頂点: $v_1 v_3 v_6$

U:未調査頂点: v_2 v_4 v_5

U から最小の distance を持つ頂点を選ぶ

プリム法 - ステップ 4 -



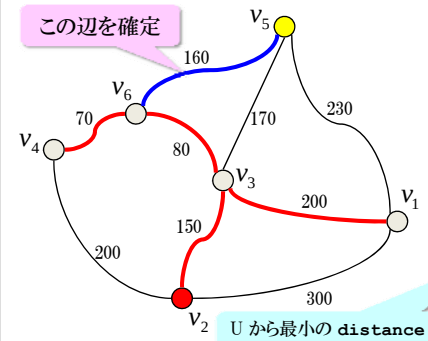
f1	0
f2	150
f3	200
f4	70
f5	160
f6	80

V:調査済頂点: $v_1 v_3 v_6$ v_4

U:未調査頂点: v_2 v_5

U から最小の distance を持つ頂点を選ぶ

プリム法 - ステップ 5 -



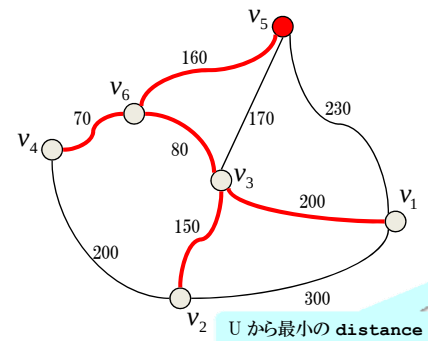
f1	0
f2	150
f3	200
f4	70
f5	160
f6	80

V:調査済頂点: $v_1 v_3 v_6$ $v_4 v_2$

U:未調査頂点: v_5

U から最小の distance を持つ頂点を選ぶ

プリム法 - 終了 -



f1	0
f2	150
f3	200
f4	70
f5	160
f6	80

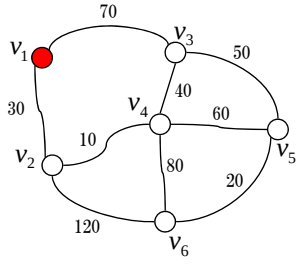
V:調査済頂点: $v_1 v_3 v_6 v_4 v_2 v_5$

U:未調査頂点:

U から最小の distance を持つ頂点を選ぶ

演習：プリム法

- 最小木を完成させよ



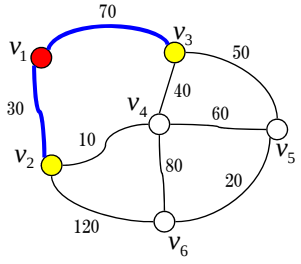
f1	0
f2	∞
f3	∞
f4	∞
f5	∞
f6	∞

V:調査済頂点:

U:未調査頂点: v_1 v_2 v_3 v_4 v_5 v_6

演習：プリム法 -ステップ 1-

- 最小木を完成させよ



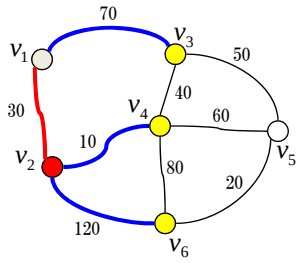
f1	0
f2	30
f3	70
f4	∞
f5	∞
f6	∞

V:調査済頂点: v_1

U:未調査頂点: v_2 v_3 v_4 v_5 v_6

演習：プリム法 -ステップ 2-

- 最小木を完成させよ



f1	0
f2	30
f3	70
f4	10
f5	∞
f6	120

V:調査済頂点
v_1 v_2

U:未調査頂点
v_3 v_4 v_5 v_6

演習：プリム法 -ステップ 3-

- 最小木を完成させよ

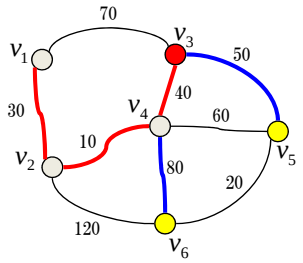
f1	0
f2	30
f3	40
f4	10
f5	60
f6	80

V:調査済頂点
v_1 v_2 v_4

U:未調査頂点
v_3 v_5 v_6

演習：プリム法 -ステップ 3-

- 最小木を完成させよ



f1	0
f2	30
f3	40
f4	10
f5	50
f6	80

V:調査済頂点
v_1 v_2 v_4 v_3

U:未調査頂点
v_5 v_6

演習：プリム法 -ステップ 4-

- 最小木を完成させよ

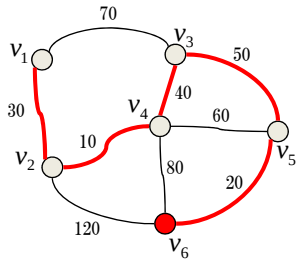
f1	0
f2	30
f3	40
f4	10
f5	50
f6	20

V:調査済頂点
v_1 v_2 v_4 v_3 v_5

U:未調査頂点
v_6

演習：プリム法 -終了-

- 最小木を完成させよ



f1	0
f2	30
f3	40
f4	10
f5	50
f6	20

V:調査済頂点
v_1 v_2 v_4 v_3 v_5 v_6

U:未調査頂点

第 10 講のまとめ

- グラフアルゴリズムの紹介
  - 最短路の問題
    - ダイクストラのアルゴリズム
  - 最小木の問題
    - プリムのアルゴリズム

第 11 講の予告

- 1月9日（金）

- 予習
  - 時間があれば以下の語句を図書館，インターネットなどで調べてみよう
  - Google や Wikipedia で調べれば良い

  - バックトラック法
  - 分枝限定法
  - 動的計画法
  - 貪欲法
  - NP完全問題