アルゴリズムとデータ構造II

講義4：

加重グラフ

https://elms.u-aizu.ac.jp

加重グラフ

- A 加重グラフ グラフです *G*（*V*、*E*）実数値で各エッジに割り当てられた重み。同様に、加重グラフはトリプルです*G*（*V*、*E*、*W*）、どこ *V*頂点のセットです、*E*はエッジのセットであり、*W*の要素をマッピングする関数です *E*実数に。ザ・重量 エッジ上も呼ばれます 距離 または 費用。

距離行列

- 加重グラフ G（V、E、W）で表すことができます
  距離行列

$$D_{n\times\times n}、n=|V|、$$

どこ

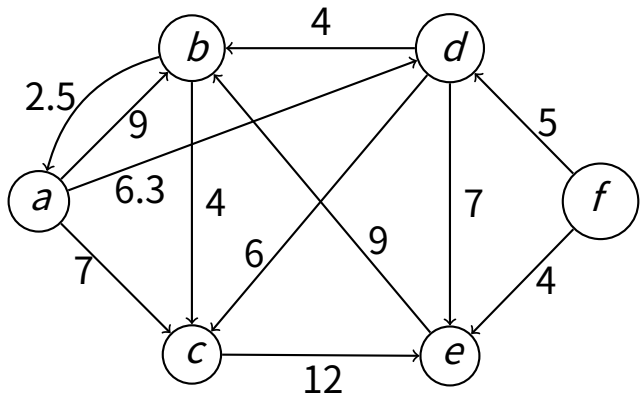
$$D[i、i]=0、$$

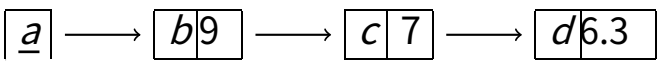
と1のために $\leq$ 私 ／ $=j\leq n$、エッジの場合（i、j）$\in E$その後 $D[i、j]$
（の重みですi、j）、さもないと $D[i、j]$無限です ∞
（実際には十分な数）。

グラフ　表現

	a	b	c	d	e	f
a	0	9	7	6.3	M	M
b	2.5	0	4	M	M	M
c	M	M	0	M	12	M
d	M	4	6	0	7	M
e	M	9	M	M	0	M
f	M	M	M	5	4	0

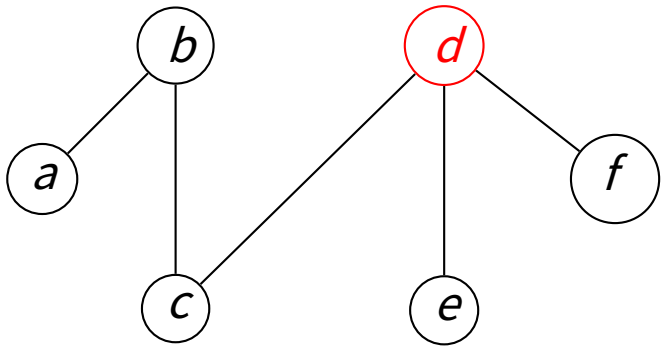
$M = \infty$





最小全域木

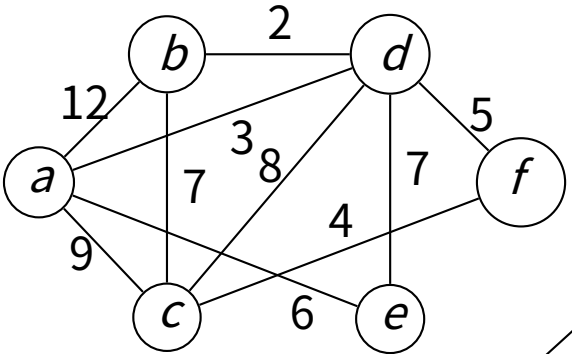
- ルートツリーは、各頂点について v、
  フォームのパスは1つだけです
  （（r、x）（x、y）、。。。、（z、v）、ど 特別な頂点
  こ rは木の根と呼ばれます。



サブグラフ、スパニングツリー

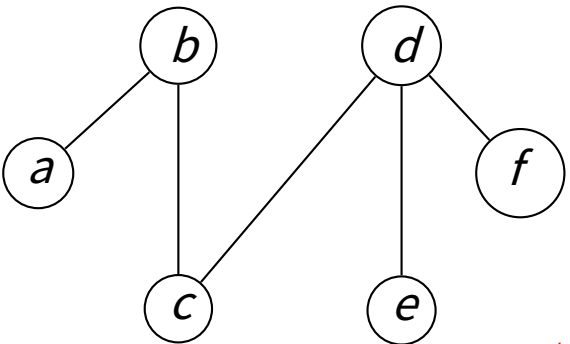
- A サブグラフ グラフの G（V、E）グラフです G'（V'、E'）
  そのような $V' \subseteq V$ そして $E' \subseteq E$。
- A スパニングツリー グラフの G（V、E）サブグラフです
  テレビ（V'、E'）そのような T 根付いた木であり、 $V' = V$。
- グラフのスパニングツリーは、次のようにして見つけることができます。 DFS または
  BFS。

加重グラフとその スパニングツリー



加重グラフ G

のスパニングツリー G



最小スパニングツリー

- しましょう $テレビ、E)$ 重み付きグラフのスパニングツリーである

  Gそして

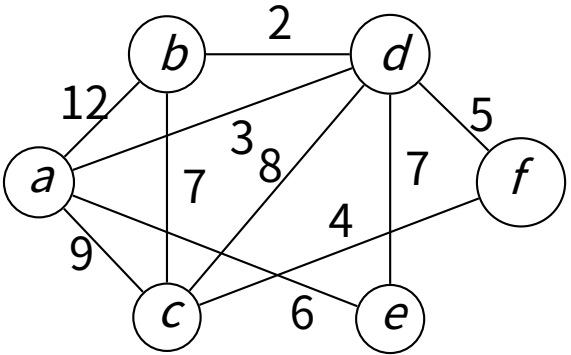
$$W（T）= \sum_{（v、w）\in E'} W（v、w）$$

  のエッジの重みの合計になります T、どこ $W（v、w）$
  エッジの重みを示します（v、w）。

- A 最小スパニングツリー（MST） の Gスパニングです
  木 T'の Gそのような

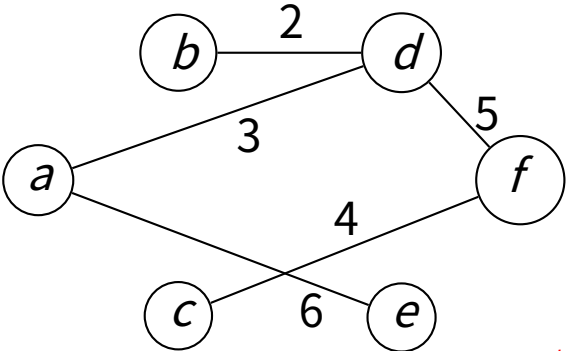
$$W（T）=\min\{W（T）\mid T\text{のスパニングツリーです }G\}。$$

加重グラフとそのMST



加重グラフ G

最小
スパニング の木 G



MSTの例

- 最小全域木問題には多くの問題があります
  電力ネットワークや電話ネットワークの構築などのアプリケーション。

MSTに関する注記

- 複数のMSTが存在する可能性があります（同じ重量）；
- Gが接続されている場合、すべての頂点がMSTにあります。
  それ以外の場合は、最小全域木があります。
- ナイーブアプローチ：すべてのスパニングツリーをリストし（どのように？）
  重みを計算します。次に、最小値を見つけます。
  非効率的...

プリムのアルゴリズム

1.1。 任意の頂点を選択します r の G（V、E）の最小全域木のルートとして G。 部分解（スパニングツリー）を想定します T 取得されました（最初は、 $T = \{r\}$）。

2.2。 エッジを選択します（v、w）そのような $v \in T$、$w \in V - T$、とエッジの重み（v、w）のノードからのエッジの最小値です T のノードへ $V - T$。

3.3。 ノードを追加します w に T。

4.4。 まで上記のプロセスを繰り返します $T = V$。

プリムのアルゴリズムの重要なアイデア

- アルゴリズムの要点は、どのように決定するかです
  はしっこ（v、w）最小重量で。
- エッジの場合（v、w）、しましょう D[v、w]（の重みであるv、w）。
  各ノードに対してwin V−T、しましょう
  $d[w] = \min\{D[v、w] \mid v \in T\}$。
- 場合 wのどのノードにも隣接していません T、$d[w] = \infty$。
  当初、 $T = \{r\}$とのために $w \neq r$、$d[w] = D[r、w]$。

プリムのアルゴリズムの重要なアイデア2

- 新しい頂点のとき c に追加されます T、 にとって

  $w \in V - T$、$d[w]$ の最小値に更新されます $d[w]$

  そして $D[c、w]$、 すなわち、 $d[w] = \min\{d[w]、D[c、w]\}$。

- 追加の配列adj $[w]$ それぞれに使用されます $w \in V - T$

  ノードを示すため v に T そのような $D[v、w] = d[w]$。

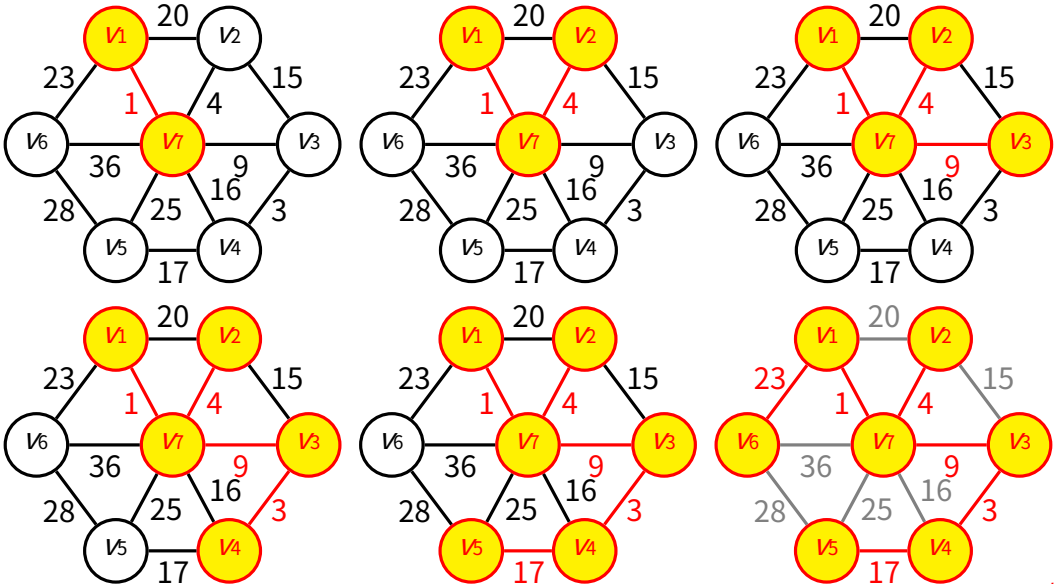
- したがって、プリムのアルゴリズムは次のように実装できます。

  続きます。

プリムのアルゴリズム

```
X = ∅; T = {r} / * r ルートです。 * /
すべてのために w ∈ V − T）{d [w] = D [r、 w]; adj [w] = r;}
にとって （i = 1; i <| V |; i ++）{
    検索 v ∈ V − T そのような d [v] = min {d [x] | バツ∈ V − T};
    /*追加 （adj [v]、 v） 木までの距離を調整します* /
    X = X ∪{（adj [v]、 v）}; T = T ∪{v};
    for（すべてのノード w ∈ V − T に隣接 v）{
        if（D [v、 w] <d [w]）{
            d [w] = D [v、 w]; adj [w] = v;}
    }
}
```

プリムのアルゴリズム



プリムのアルゴリズムの複雑さ

- グラフが隣接する（距離）で表される場合
  行列、プリムのアルゴリズムの時間計算量は
  $O(|V|^2)$。
- プリムのアルゴリズムは、次の方法でより効率的に作成できます。
  隣接リストを使用してグラフを維持し、ノードの優先キューを維持します。 T。 この実装では、プリムのアルゴリズムの時間計算量は次のようになります。 $O((|V|+|E|)$ ログ$|V|)$。

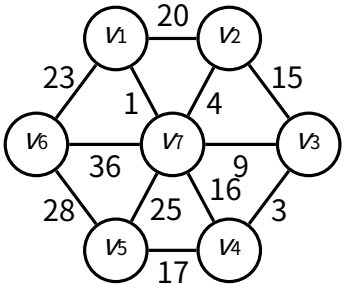
クラスカルのアルゴリズム

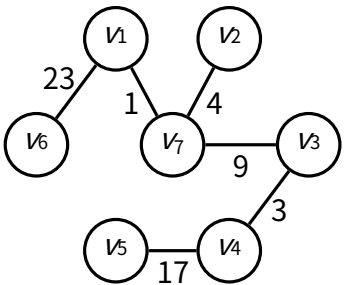
- クラスカルのアルゴリズムは、
  グラフの最小全域木。
- アルゴリズムは、次のようにして最小全域木を見つけます。
  を選択する エッジ 重みが小さい順に、エッジがサイクルを引き起こさない場合は、エッジをツリーに含めます。
- アルゴリズムでは、 Q は優先キュー（ヒープ）です。

クラスカルのアルゴリズム

$X = \{\{v\} \mid v \in V\}$; $T = \varnothing$;
のエッジを構築します E 最小ヒープに Q;
一方（Q ／$=\varnothing$）{
　エッジを削除します（v、w）からの最小重量の Q
　　ヒープ状態を復元します;
　if（$v \in V_{私}$ そして $w \in V_j$ と $V_{私}$ ／$= V_j$ そして $V_{私}$、$V_j \in$ バツ）{
　　交換 $V_{私}$ そして V_j に バツ沿って $V_{私} \cup V_j$;
　　$T = T \cup \{$（v、w）$\}$;
　}
}

クラスカルのプロセス





縁	アクション	
		{{v1}、{v2}、{v3}、{v4}、{v5}、{v6}、{v7}}（接続されたコンポーネント）
（v1、v7）	追加	{{v1、v7}、{v2}、{v3}、{v4}、{v5}、{v6}
（v3、v4）	追加	{{v1、v7}、{v2}、{{v3、v4}、{v5}、{v6}
（v2、v7）	追加	{{v1、v2、v7}、{{v3、v4}、{v5}、{v6}
（v3、v7）	追加	{{v1、v2、v3、v4、v7}、{v5}、{v6}
（v2、v3）	拒否する	
（v4、v7）	拒否する	
（v4、v5）	追加	{{v1、v2、v3、v4、v5、v7}、{v6}
（v1、v2）	拒否する	
（v1、v6）	追加	{{v1、v2、v3、v4、v5、v7、v6}

クラスカルのアルゴリズムの複雑さ

- クラスカルのアルゴリズムの時間計算量は

  $O(|E|\log|E|)$ 隣接リストとプライオリティキューが使用されている場合。

Boruvkaのアルゴリズム

- Boruvkaのアルゴリズム（1926）：おそらくコンピューター実装のための3つの古典的なアルゴリズムの中で最も簡単なものです（複雑なデータ構造は必要ありません）。プリムのアルゴリズムと同様の手順を実行しますが、グラフ全体で並行して実行されます。

- 同時に、Boruvkaのアルゴリズム：
  1.1。リストを作る L の n 木、それぞれが単一の頂点。
  2.2。一方（L 複数の木があります）
    2.1 それぞれについて T に L、接続する最小のエッジを見つける T に $G- T;$
    2.2 これらすべてのエッジをMSTに追加します（マージ）。
- 時間計算量：$O（|E|$ログ$|V|）$
- BoruvkaとPrimのハイブリッド：$O（|E|$ログログ$|V|）$

Maggs＆Plotkin Algorithm（1994）

- 加重グラフ G（V、E、W）、$|V|=n$

  初期化：$d[i、j]^{(0)}=d[i、j]$、$1 \leq i、j \leq n$;

  にとって $k=1$ に n 行う

  　にとって $i=1$ に n 行う

  　　にとって $j=1$ に n 行う

  　　$d[i、j]^{(k)}=\min\{d[i、j]^{(k-1)}、$

  　　$\max(d[i、k]^{(k-1)}、d[k、j]^{(k-1)})\}$;

- エッジ $d[i、j]^{(n)}=d[i、j]^{(0)}$ MSTからです
- 複雑さは $O(|V|^3)$