データ構造とアルゴリズムII

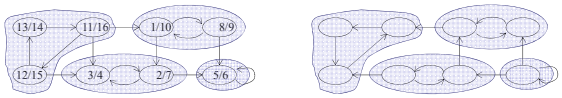
第8回
強連結成分／最小全域木

予定 (II)

6/13: 第7回：幅優先，深さ優先探索，トポロジカルソート
6/20:第8回：強連結成分分解，最小全域木
6/27:小テスト
7/4: 第9回：単一始点最短路 (I)
7/11: 第10回：単一始点最短路 (II)，全点対最短経路
7/18: 第11回：最大フロー
7/25: 定期試験

強連結成分

- 深さ優先探索の古典的な応用：有向グラフを強連結成分に分割
- 有向グラフG=(V,E)の強連結成分とは，Cのすべての頂点対u, vに対してuとvがお互いに到達可能な頂点の極大集合$C \subseteq V$

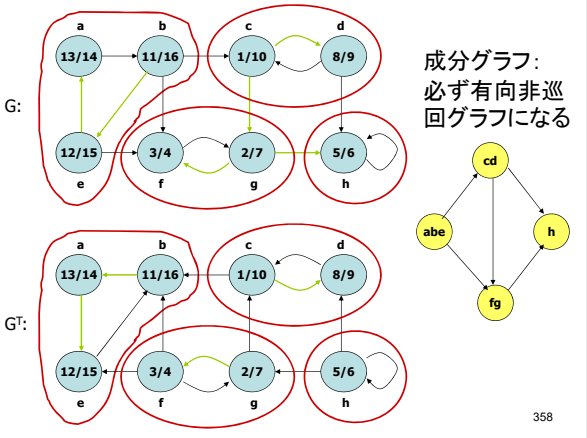


- 次に示すアルゴリズムは深さ優先探索をG, G^T（辺の向きを逆転したグラフ）に関して
実行することにより，有向グラフGの強連結成分を
Θ(V+E)時間で求める．

強連結成分を求めるアルゴリズム

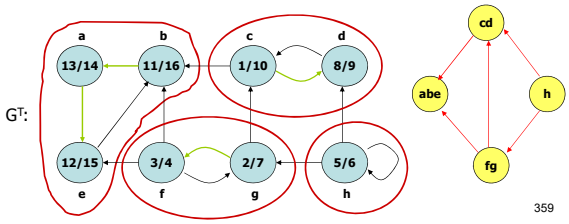
STRONGLY-CONNECTED-COMPONENTS(G)
1 DFS(G)を呼び出して，各頂点uに対して
終了時刻 u.f を計算する
2 G^Tを計算する
3 DFS(G^T)を呼び出すが，DFSのメインループでは u.f の
降順に頂点を探索する
4 第3ステップの深さ優先探索のそれぞれの木の頂点を，それぞれ分離された強連結成分として出力する

以下，このアルゴリズムの正しさを示す．

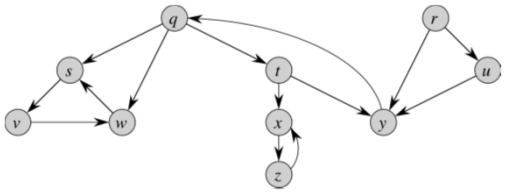


直感的なイメージ

- 成分グラフをトポロジカルソートしたものを考える．ソート順で最初にあるものが，終了時刻が遅い．
- ソート順で，最初にあるものからG^Tに関して深さ優先探索を行うと，まだ探索していない強連結成分には到達不可能．
よって，強連結成分のみで木を構成．

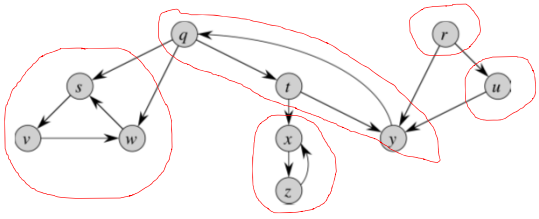


演習：強連結成分



– 上のグラフを強連結成分に分解せよ．最初の深さ優先探索において，頂点の選択順／隣接リストはアルファベット順とする．

演習：強連結成分



• 上のグラフを強連結成分に分解せよ．

補題 22.13 :
C and C' を，グラフG=(V, E) の相異なる強連結成分とする．u, v in C , u', v' in C' に関して，u から u' への経路が存在すると仮定する．
この場合，v' から v への経路は存在しない．
証明：もし経路があれば，C∪C'が強連結成分になる

定義: For $U \subseteq V$, let us define

$$d(U) = \min_{u \in U} \{ u.d \}$$

$$f(U) = \max_{u \in U} \{ u.f \}$$

注意：以下，発見時刻，終了時刻は，最初の深さ優先探索の時の時刻とする．

補題 22.14
C と C' を，グラフG=(V, E) における相異なる強連結成分とする．また，u∈C, v∈C' に関して，辺 (u, v) が存在すると仮定する．
この場合，f (C) > f (C') が成立．
証明: Case (1): d(C) < d(C')の場合:
- x を，C中で最初に発見された頂点とする．
  xが発見された時刻では，C および C' 中の他の頂点は白．
- x からC中の任意の頂点への白頂点のみを経由する経路が存在．また，C' 中の任意の頂点wへも，白頂点のみを経由して，x ⇝ u → v ⇝ wで到達可能．
- C および C' 中の任意の頂点はxの子孫であるため，xの終了時刻= f (C) > f(C')．

補題 22.14
C と C' を，グラフG=(V, E) における相異なる強連結成分とする．また，u∈C, v∈C' に関して，辺 (u, v) が存在すると仮定する．
この場合，f (C) > f (C') が成立．
証明: Case (2): d(C) > d(C')の場合:
- y を，C'中で最初に発見された頂点とする．
- yが発見された時点で，C'中の他の頂点はすべて白であり，yから白頂点のみを経由して到達可能．すなわち，C'の任意の頂点はyの子孫．
- よって，yの終了時刻= f(C')．C'からCの経路は存在しない．
- よって，任意のw ∈ C の終了時刻は，yの終了時刻＝ f(C') より大きい．

系 22.15: C and C' を相異なる強連結成分とする．u∈C, v∈C'に関して，(u, v) が E^Tに含まれるとする．
この時，f (C) < f (C').

証明: (v, u) がEに含まれるので，先の補題より，
f(C') > f (C).

定理 22.16

Strongly-Connected-Component(G) は，正しく強連結成分を求める．

証明：

- G^Tで得られる深さ優先探索木の数kに関する数学的帰納法を用いる．
- k=0の時は明らかに成立．
- 最初のk個の深さ優先探索木がSCCであると仮定．
- (k+1)番目の木を考える．
  この木のルートをu とし，これが含まれるSCCをCとする．
- uの終了時刻＝ f (C) > f (C')，ただしC'は，まだ探索されていない他のSCC．（最初に実行した）終了時刻の降順で頂点を選んでいるため．

366

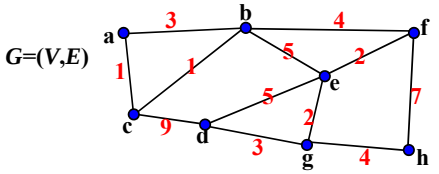
- 帰納法の仮定と系22.15から，G^T に属し，Cの任意の頂点から出る辺は，まだ探索されていない他のSCCの要素を指すことはない．
- よって，C以外のSCCの要素である頂点が，G^Tに関する探索時で，uの子孫となることはない．
- また，Cの要素は必ずG^T においてuから到達可能．
- 従って，G^T の深さ優先木で，uを根とする木は，ちょうど一つのSCCを構成する． ■

367

23. 最小全域木

368

最小全域木



- 連結重み付き無向グラフを対象
- 全域木＝すべての頂点を含む木（閉路なし）
- 最小全域木＝辺の重みの和を最小とする全域木
- 応用：通信ネットワークの構築等
- 全域木の数は指数的，効率よく最小全域木を見つけたい

369

抽象的な貪欲アルゴリズム

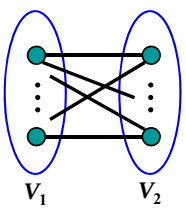
```
A = ∅;
while(T=(V,A) is not a spanning tree of G) {
  select a safe edge for A ;
}
```

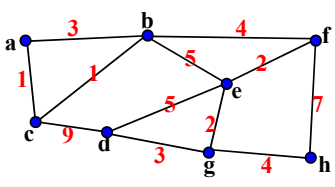
- 貪欲的に，Aに辺を追加．
- 安全な辺：Aに加えても，Aがいずれかの最小全域木の部分集合であることを保証する辺
- どうやって安全な辺を見つける?

370

MSTに関する性質

Let $V = V_1 + V_2$, $\Delta(V_1,V_2) = \{uv \mid u \in V_1 \& v \in V_2\}$. if $xy \in \Delta(V_1,V_2)$ and $w(xy) = \min\{w(uv) \mid uv \in \Delta(V_1,V_2)\}$, then xy is contained in a MST.





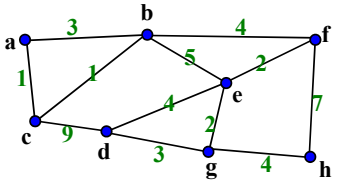
371

Kruskalのアルゴリズム

```
A= ∅;
for( each edge in order by nondecreasing weight )
  if( adding the edge to A doesn't create a cycle ){
      add it to A;
      if( |A| == n–1 ) break;
  }
```

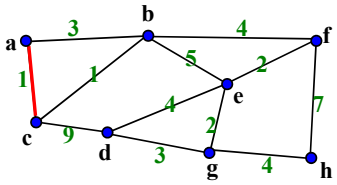
- Aは森を構成(隣接する辺がAに含まれない頂点は，単独で木を構成すると考える)．
- 重みが最小の辺 e を選ぶ．
- eが，A中の同じ木に属する頂点を結ぶか否かをチェック，そうでなければ，eはsafe: eが二つの木T1, T2を結ぶなら，V1をT1中の頂点，V2を残りとすれば前述の性質が成立 372

Kruskalのアルゴリズムの実行例



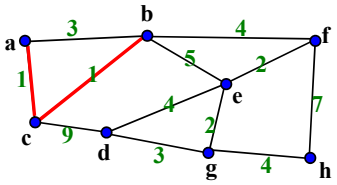
373

Kruskalのアルゴリズムの実行例



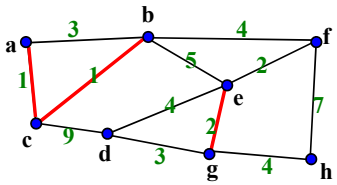
374

Kruskalのアルゴリズムの実行例



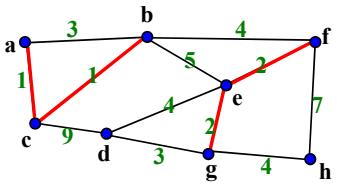
375

Kruskalのアルゴリズムの実行例



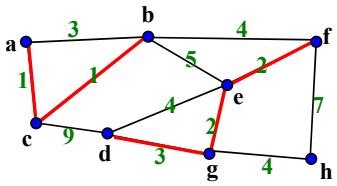
376

Kruskalのアルゴリズムの実行例

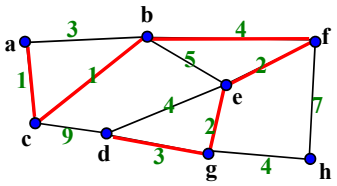


377

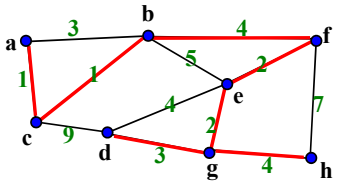
Kruskalのアルゴリズムの実行例



Kruskalのアルゴリズムの実行例



Kruskalのアルゴリズムの実行例



MST cost = 17

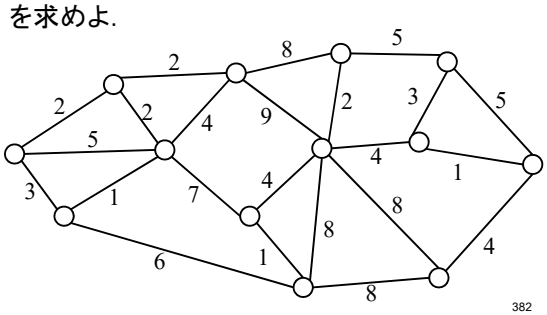
Kruskalのアルゴリズム

```
A = ∅;   initial(n); // for each node x construct a set {x}
for( each edge xy in order by nondecreasing weight)
  if ( ! find(x, y) ) {
     union(x, y);
     add xy to A;
     if( |A| == n–1 ) break;
  }
```

find(x, y) = true iff. x and y are in the same set
union(x, y): unite the two sets that contain x and y, respectively.

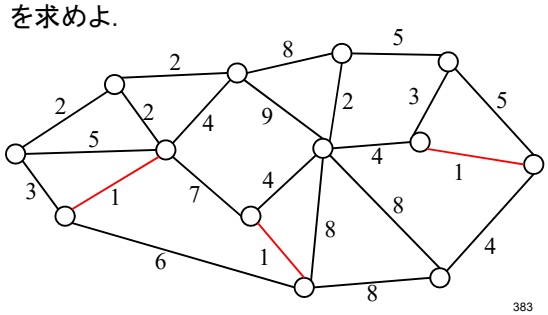
演習：Kruskalのアルゴリズム

- Kruskalのアルゴリズムを用いて，以下のMSTを求めよ．



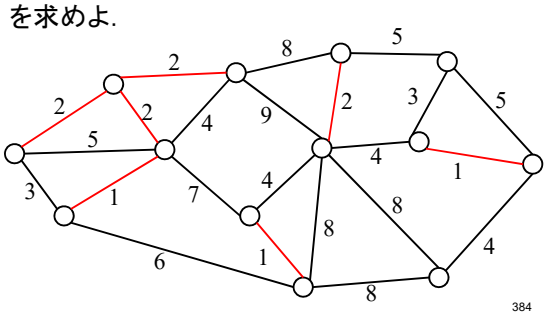
演習：Kruskalのアルゴリズム

- Kruskalのアルゴリズムを用いて，以下のMSTを求めよ．



演習：Kruskalのアルゴリズム

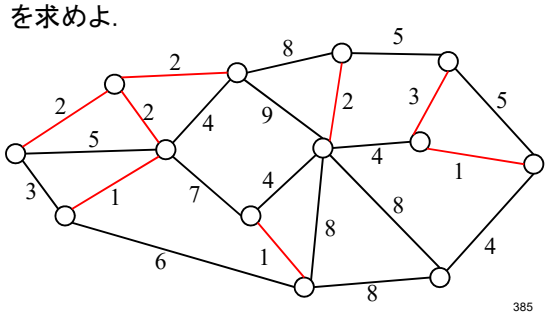
- Kruskalのアルゴリズムを用いて，以下のMSTを求めよ．



384

演習：Kruskalのアルゴリズム

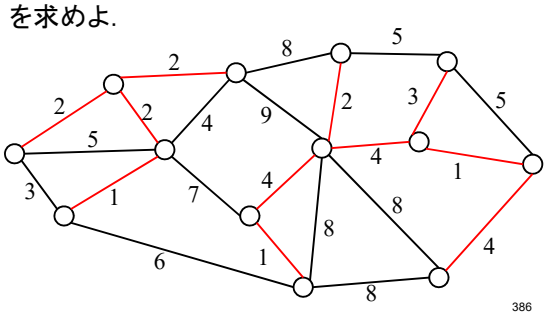
- Kruskalのアルゴリズムを用いて，以下のMSTを求めよ．



385

演習：Kruskalのアルゴリズム

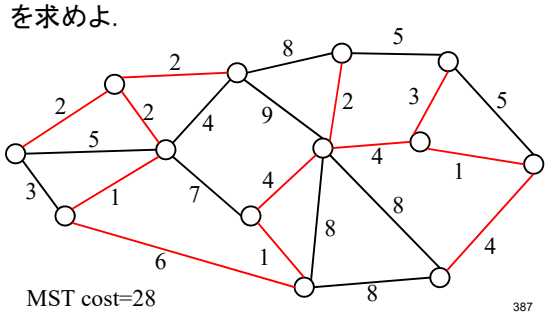
- Kruskalのアルゴリズムを用いて，以下のMSTを求めよ．



386

演習：Kruskalのアルゴリズム

- Kruskalのアルゴリズムを用いて，以下のMSTを求めよ．



MST cost=28

387

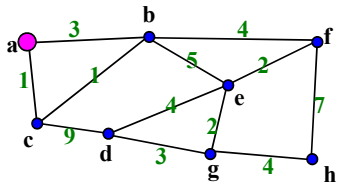
Primのアルゴリズム

ALGORITHM *Prim*(*G*)
// Input: A weighted connected graph $G=(V,E)$
// Output: A MST $T=(V, A)$
$V_T \leftarrow \{v_0\}$ // Any vertex will do;
$A \leftarrow \emptyset$;
for $i \leftarrow 1$ to $|V|-1$ do
find an edge $xy \in \Delta(V_T, V-V_T)$ **s.t. its weight is minimized among all edges in** $\Delta(V_T, V-V_T)$**;**
$V_T \leftarrow V_T \cup \{y\}$; $A \leftarrow A \cup \{xy\}$;

- 実装上のポイント：V_T と $V-V_T$ を結ぶ最小重みの辺を効率よく見つける．

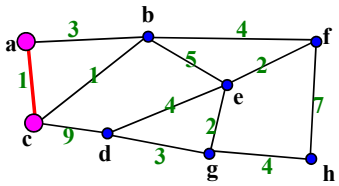
388

Primのアルゴリズムの実行例



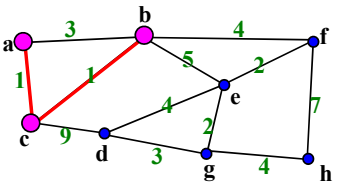
389

Primのアルゴリズムの実行例



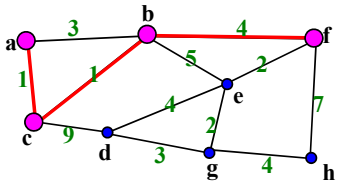
390

Primのアルゴリズムの実行例



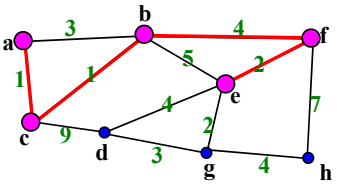
391

Primのアルゴリズムの実行例



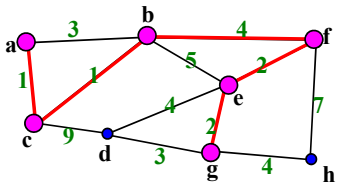
392

Primのアルゴリズムの実行例



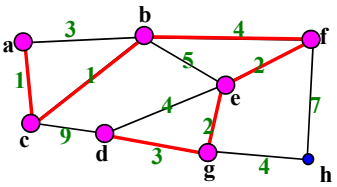
393

Primのアルゴリズムの実行例



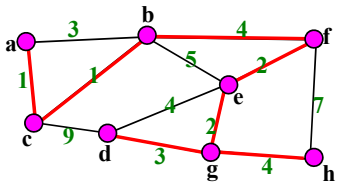
394

Primのアルゴリズムの実行例



395

Primのアルゴリズムの実行例



MST cost = 17

396

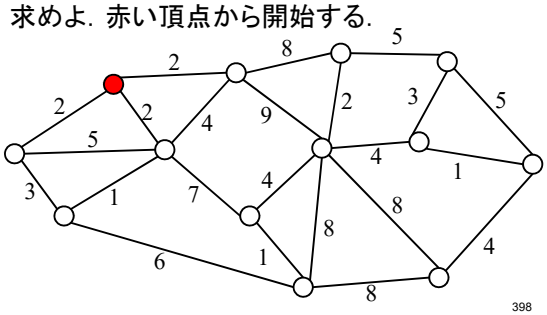
Primのアルゴリズム

```
Built a priority queue Q for V with key[u] = ∞ ∀ u∈V;
key[v0] = 0;  π[v0] = Nil; // Any vertex will do
V_T = ∅
While (Q ≠ ∅) {
   u = Extract-Min(Q);
   add u to V_T ;
   for( each v ∈ Adj(u) )
      if (v ∈ Q && w(u, v) < key[v] ) {
         π[v] = u;
         key[v] = w(u, v);
         Change-Priority(Q, v, key[v]);
      }
}
```

397

演習：Primのアルゴリズム

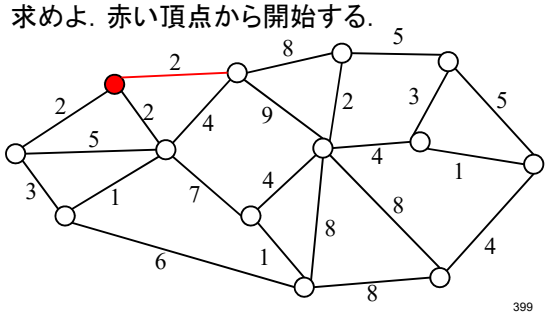
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



398

演習：Primのアルゴリズム

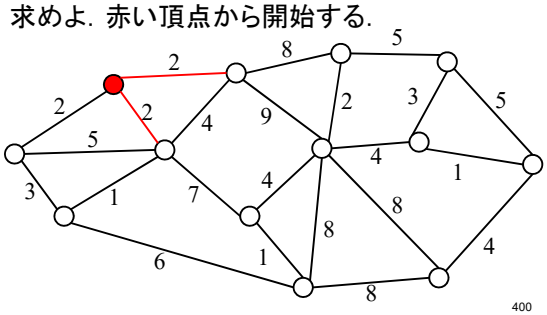
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



399

演習：Primのアルゴリズム

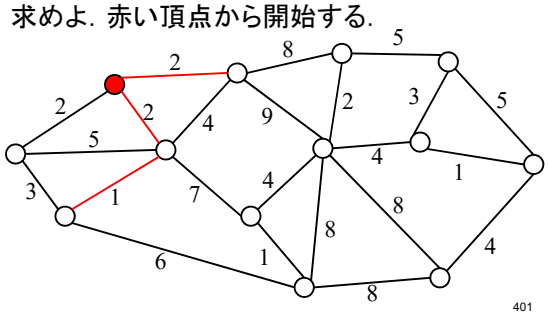
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



400

演習：Primのアルゴリズム

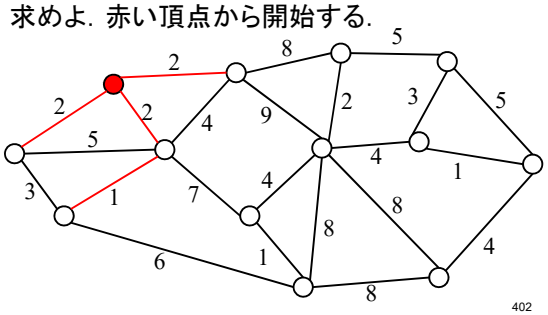
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



401

演習：Primのアルゴリズム

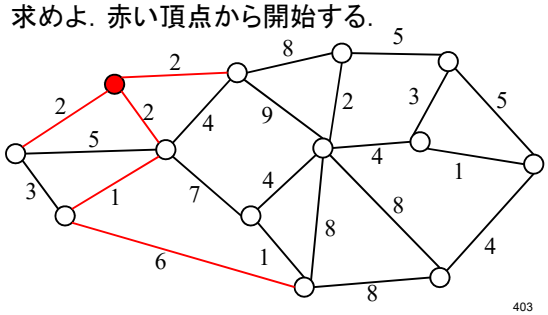
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



402

演習：Primのアルゴリズム

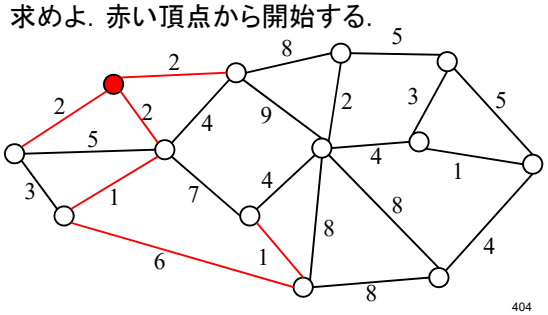
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



403

演習：Primのアルゴリズム

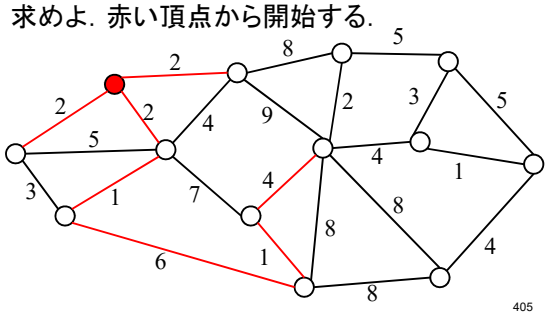
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



404

演習：Primのアルゴリズム

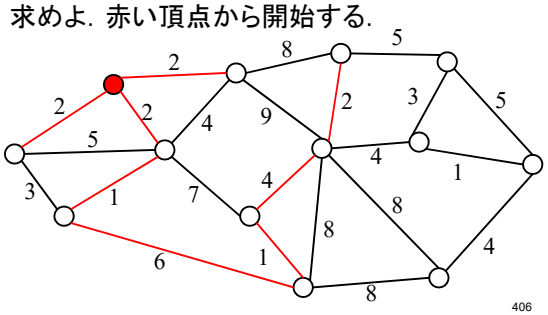
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



405

演習：Primのアルゴリズム

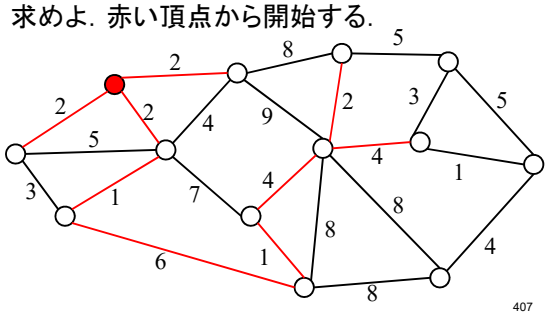
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



406

演習：Primのアルゴリズム

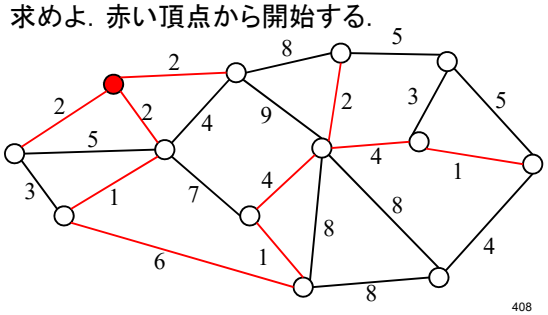
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



407

演習：Primのアルゴリズム

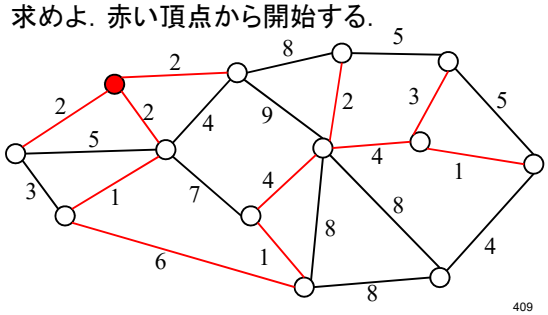
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



408

演習：Primのアルゴリズム

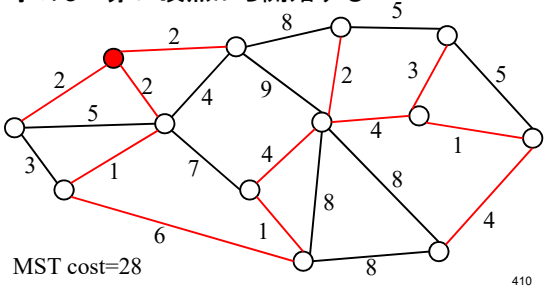
- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



409

演習：Primのアルゴリズム

- Primのアルゴリズムを用いて，以下のMSTを求めよ．赤い頂点から開始する．



MST cost=28

410

MSTアルゴリズムの分析

- $n = |V(G)|$, $m = |E(G)|$ とする．
- Kruskalのアルゴリズムの実行時間：
  $O(m \log m) = O(m \log n)$ ：ソートの計算量
- Primのアルゴリズムの実行時間：
- ヒープ：
  $O((m+n) \log n) = O(m \log n)$ ：extract-minがn回，priorityの修正がm回．
  - もう少し凝ったヒープ（フィボナッチヒープ）：
    $O(n \log n + m)$

411