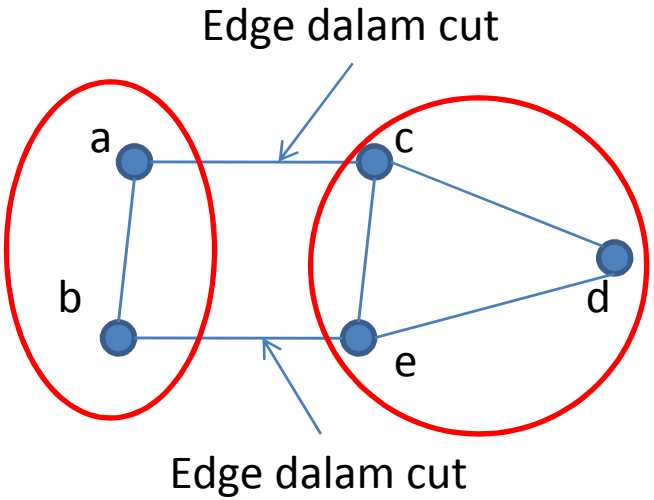
Greedy algorithms

wijanarto

Materi

- Graph
- MST
- Kruskal
- Prim
- Dijkstra

Algoritma Prim utk MST



G

Suatu potongan (CUT) dlm graf G yang di bagi mjd himpunan vertek menjadi dua part/bagian

Jadi himp. Verteknya adalah {a,b} dan {c d e}

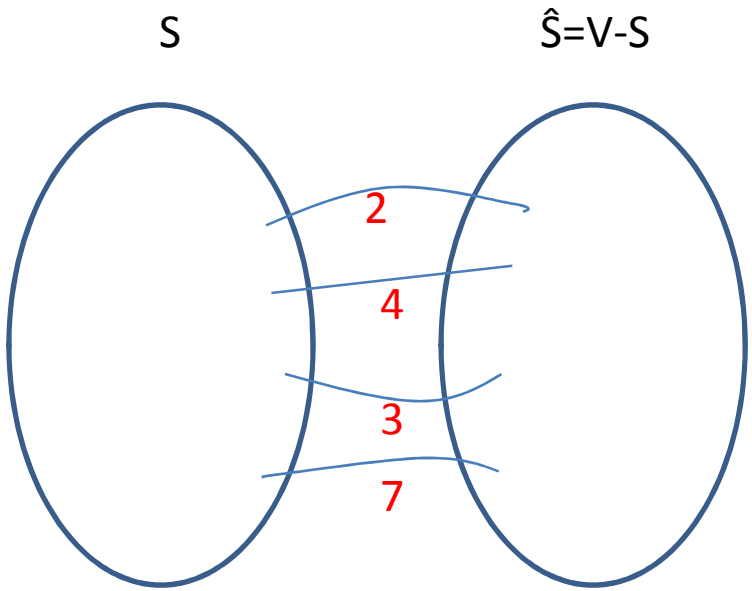
Terdapat kemungkinan banyak part/bagian dalam G, misal

{a,c,d} dan {b,e}
{c} dan {a,b,d,e}

Berapa jumlah CUT /Potongan yang berbeda yang mungkin dalam suatu G ? $2^{n-1}-1$

Algoritma Prim utk MST

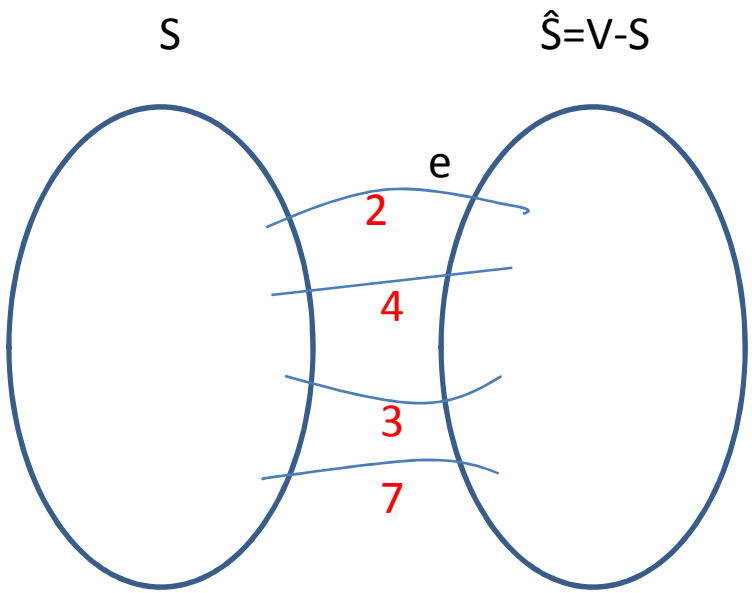
- Misal kita ambil 2 potongan dari G=(V,E) terdahulu
- Sekarang kita lihat tiap edge dalam CUT tsb
- Kita akan menghitung MST, dan misal edge tsb mempunyai bobot yang berbeda



Algoritma Prim utk MST

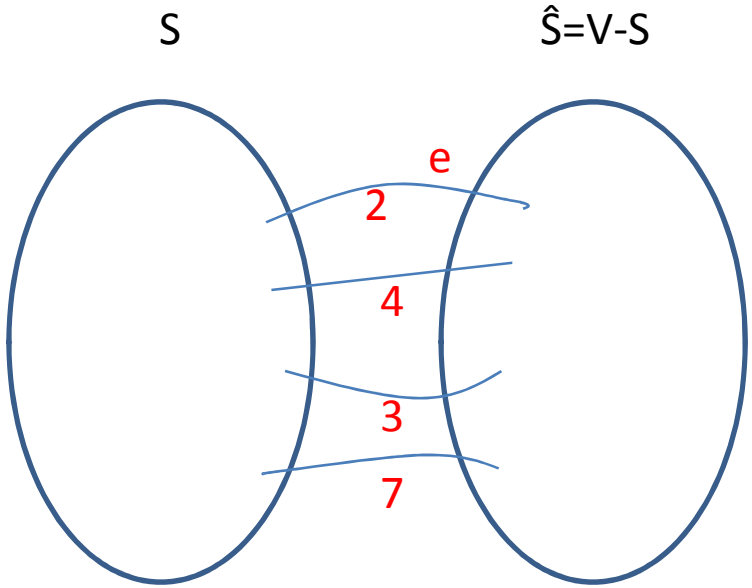
- Claims : Untuk setiap CUT (S,Ŝ), edge yang minimum dalam CUT Pasti kepunyaan MST
- Bukti dg Kontradiksi :

  Di berikan T dari suatu MST yg tidak mengandung edge e



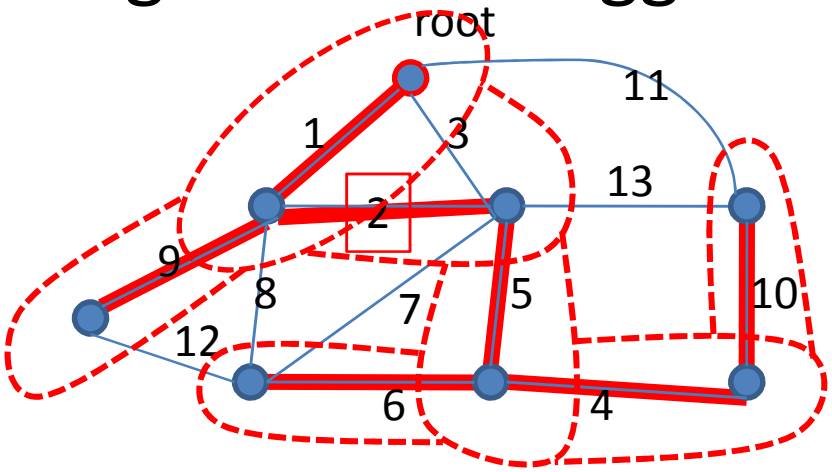
Algoritma Prim utk MST

- Tambahkan e ke T
- Penambahan e ke T membentuk cycle C
- C mengandung setidaknya satu edge (kecuali e) dalam CUT
- Akibatnya C mengandung suatu edge dg panjang/bobot lebih dari panjang/bobot e
- Dengan menghilangkan edge ini dari T∪{e}, kita dapatkan TREE yang lebih kecil



Algoritma Prim utk MST

- Bagaimana Menggunakan Claims



Membuat partisi (lingkaran garis putus), Sehingga partisi lainnya adalah selain edge berbobot 1 dan CUT edge yang Terbentuk ada banyak (9,8,2,3,11)

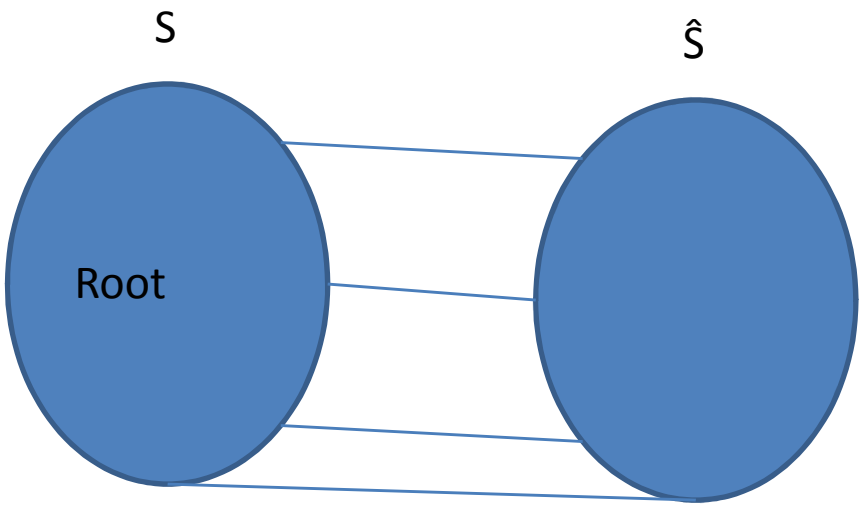
Kebetulan yang terkecil bobotnya adalah 2, jadi kita tahu edge ini seharusnya menjadi Bagian dari MST, dan partisi berubah.

Perubahan partisi ini menyebabkan edge CUT juga berubah (9,8,7,5,13,11)

Dan kemudian edge CUT yg terkecil adalah 5, Selanjutnya kita bentuk partisi baru dengan edge Yang berbeda pula begitu selanjutnya

Algoritma Prim utk MST

- Menulis psuedo code Prim utk MST



1. Mencari seluruh edge CUT dengan Bobot/panjang minimum

Struktur Data HEAP

-

Procedure	Binary heap (worst-case)	Binomial heap (worst-case)	Fibonacci heap (amortized)
MAKE-HEAP	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
INSERT	$\Theta(\lg n)$	$O(\lg n)$	$\Theta(1)$
MINIMUM	$\Theta(1)$	$O(\lg n)$	$\Theta(1)$
EXTRACT-MIN	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$
UNION	$\Theta(n)$	$O(\lg n)$	$\Theta(1)$
DECREASE-KEY	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$
DELETE	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$

Struktur Data HEAP

- BINOMIAL-HEAP-MINIMUM(H)

```
1  y ← NIL
2  x ← head[H]
3  min ← ∞
4  while x ≠ NIL
5        do if key[x] < min
6              then min ← key[x]
7                   y ← x
8             x ← sibling[x]
9  return y
```

Struktur Data HEAP

```
BINOMIAL-LINK(y, z)
1  p[y] ← z
2  sibling[y] ← child[z]
3  child[z] ← y
4  degree[z] ← degree[z] + 1
```

Struktur Data HEAP

-

```
BINOMIAL-HEAP-UNION(H1, H2)
 1  H ← MAKE-BINOMIAL-HEAP()
 2  head[H] ← BINOMIAL-HEAP-MERGE(H1, H2)
 3  free the objects H1 and H2 but not the lists they point to
 4  if head[H] = NIL
 5     then return H
 6  prev-x ← NIL
 7  x ← head[H]
 8  next-x ← sibling[x]
 9  while next-x ≠ NIL
10        do if (degree[x] ≠ degree[next-x]) or
                (sibling[next-x] ≠ NIL and degree[sibling[next-x]] = degree[x])
11              then prev-x ← x                                  ▷ Cases 1 and 2
12                   x ← next-x                                  ▷ Cases 1 and 2
13              else if key[x] ≤ key[next-x]
14                      then sibling[x] ← sibling[next-x]        ▷ Case 3
15                           BINOMIAL-LINK(next-x, x)            ▷ Case 3
16                      else if prev-x = NIL                     ▷ Case 4
17                              then head[H] ← next-x            ▷ Case 4
18                              else sibling[prev-x] ← next-x    ▷ Case 4
19                           BINOMIAL-LINK(x, next-x)            ▷ Case 4
20                           x ← next-x                          ▷ Case 4
21           next-x ← sibling[x]
22  return H
```

Struktur Data HEAP

```
BINOMIAL-HEAP-INSERT(H, x)
1   H' ← MAKE-BINOMIAL-HEAP()
2   p[x] ← NIL
3   child[x] ← NIL
4   sibling[x] ← NIL
5   degree[x] ← 0
6   head[H'] ← x
7   H ← BINOMIAL-HEAP-UNION(H, H')
```

Struktur Data HEAP

BINOMIAL-HEAP-EXTRACT-MIN(H)

1 find the root x with the minimum key in the root list of H,
and remove x from the root list of H
2 $H' \leftarrow$ MAKE-BINOMIAL-HEAP()
3 reverse the order of the linked list of x's children,
and set $head[H']$ to point to the head of the resulting list
4 $H \leftarrow$ BINOMIAL-HEAP-UNION(H, H')
5 **return** x

Struktur Data HEAP

```
BINOMIAL-HEAP-DECREASE-KEY(H, x, k)
 1  if k > key[x]
 2      then error "new key is greater than current key"
 3  key[x] ← k
 4  y ← x
 5  z ← p[y]
 6  while z ≠ NIL and key[y] < key[z]
 7        do exchange key[y] ↔ key[z]
 8           ▷ If y and z have satellite fields, exchange them, too.
 9           y ← z
10           z ← p[y]
```

Struktur Data HEAP

-

BINOMIAL-HEAP-DELETE(H, x)

1 BINOMIAL-HEAP-DECREASE-KEY($H, x, -\infty$)

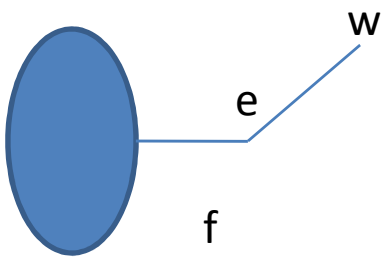
2 BINOMIAL-HEAP-EXTRACT-MIN(H)

psuedocode

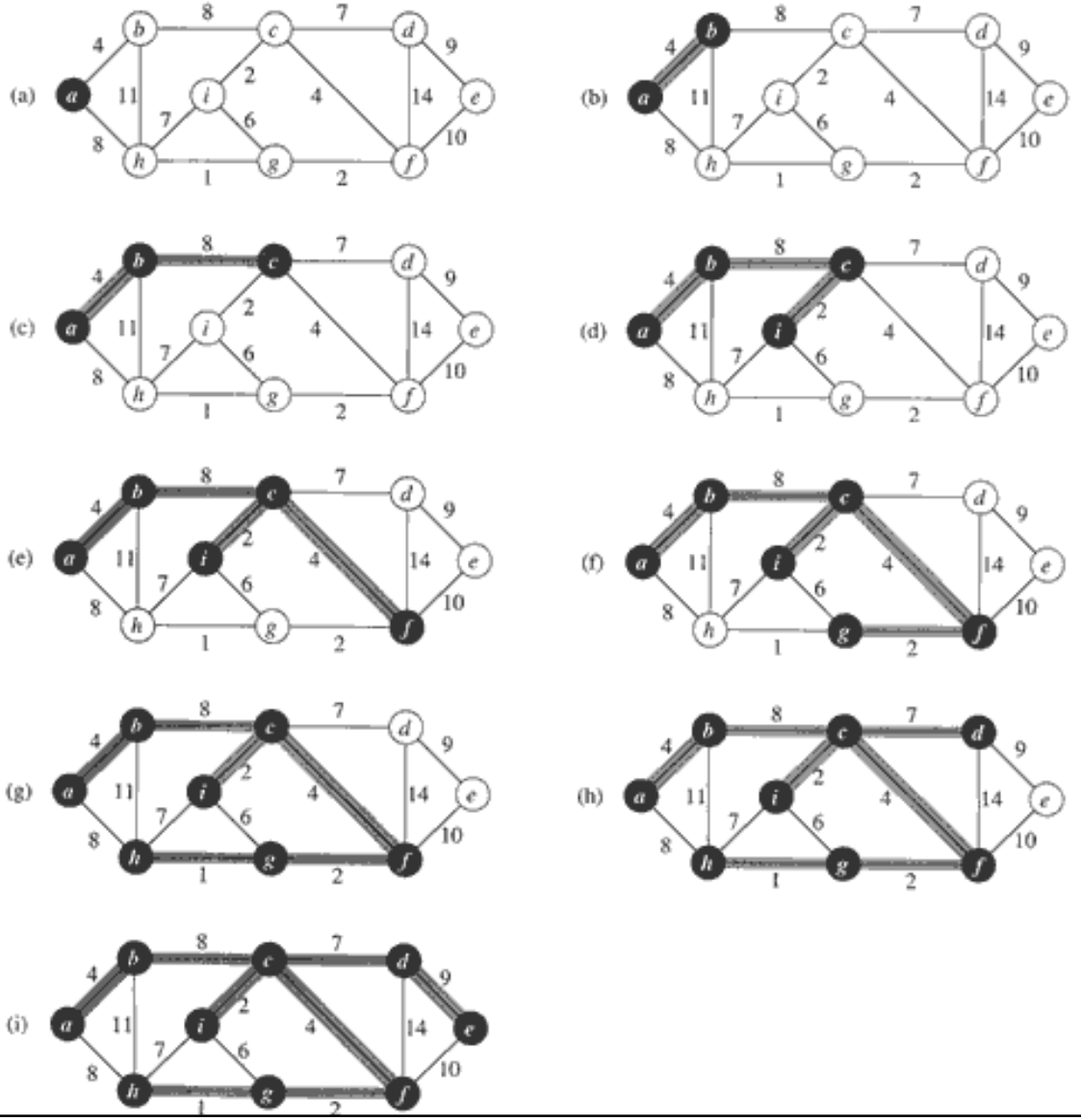
```
∀ v, S[v]=0
S[r]=1
∀  e insiden ke r do H.insert(e)
While ! H.empty() do {
        f=H.findmin() {ret edge}
        diberikan v mjd titik akhir f
        sdmk rp shg S[v]=0
        for all e adj ke v do {e=(v,w)}
                if S[w]==0 then H.insert(e)
                        else H.delete(e)
        S[v]=1
}
```

H adalah struktur data HEAP

S adalah array
S[v]=1 jika v ∈ S
0 lainnya



Contoh Lain Running Prim



Algoritma Prim (Cormen)

```
MST-PRIM(G, w, r)
 1  for each u ∈ V[G]
 2       do key[u] ← ∞
 3          π[u] ← NIL
 4  key[r] ← 0
 5  Q ← V[G]
 6  while Q ≠ ∅
 7       do u ← EXTRACT-MIN(Q)
 8          for each v ∈ Adj[u]
 9               do if v ∈ Q and w(u, v) < key[v]
10                  then π[v] ← u
11                       key[v] ← w(u, v)
```