Minimum Spanning Trees

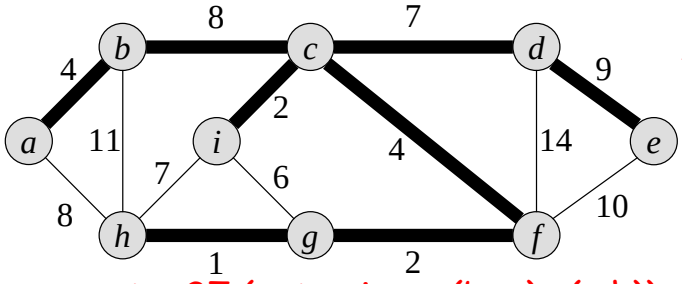
$m \geq n - 1$

Input: A connected undirected graph $G=(V, E)$

Output: A minimum spanning tree of G

{ n − 1 edges, no cycles

23-1x



{ n − 1 edges, connected

{ connected, no cycles

cost = 37 (not unique, (b, c)->(a,h))

Two greedy algorithms: Managing a set A that is always a subset of some minimum spanning tree.

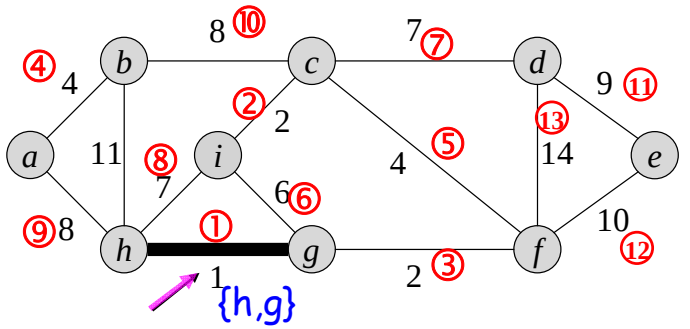
23.2 Kruskal's algorithm: smallest weighted first (no cycle)

23-1x

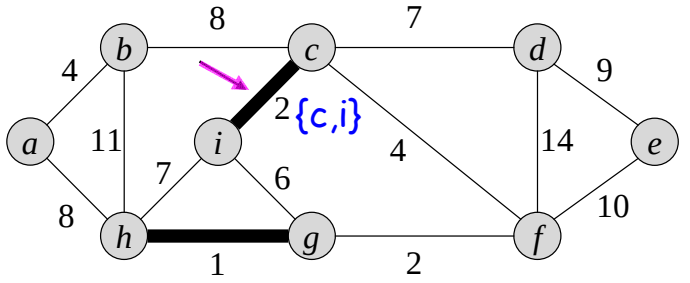
```
MST-KRUSKAL(G, w)
1  A ← ∅                                   /* tree edges */
2  for each vertex v ∈ V[G]                 } O(V)
3      do MAKE-SET(v)                      }
4  sort the edges of E into nondecreasing order by weight w      O(Elg E)
5  for each edge (u, v) ∈ E, taken in nondecreasing order by weight
6      do if FIND-SET(u) ≠ FIND-SET(v)     /* no cycle */
7          then A ← A ∪ {(u, v)}           (two ends are in different sets)
8               UNION(u, v)
9  return A
```

E × α(V) (2 FIND, 1 Union)

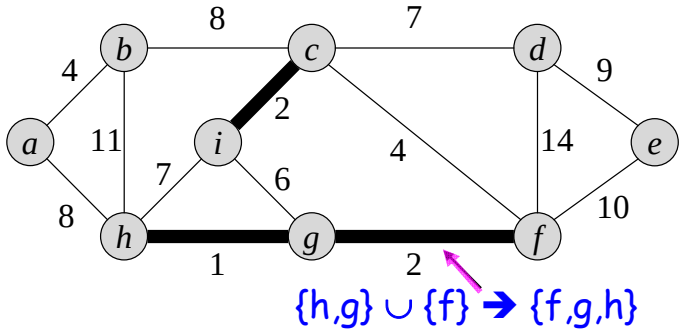
(a) 1st



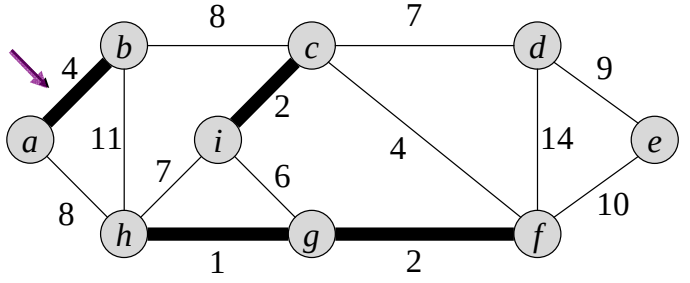
(b) 2nd



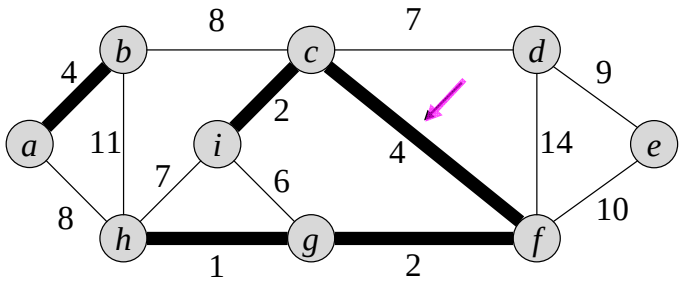
(c) 3rd



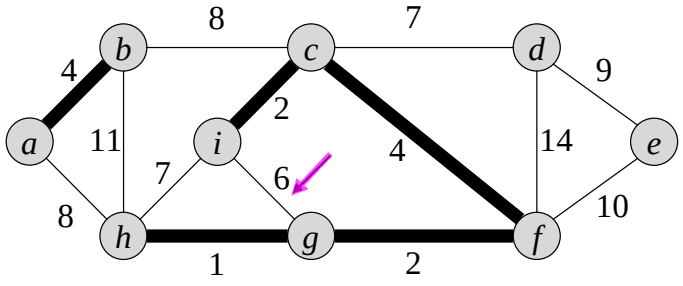
(d) 4th



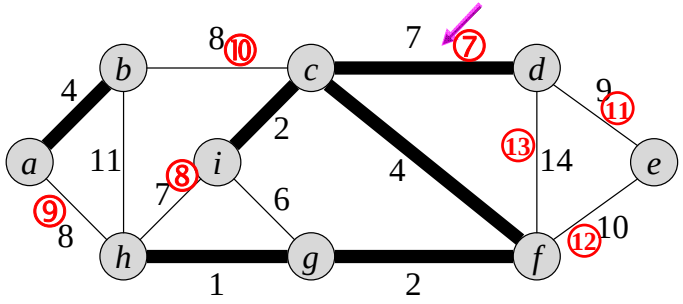
(e) 5th



(f) 6th

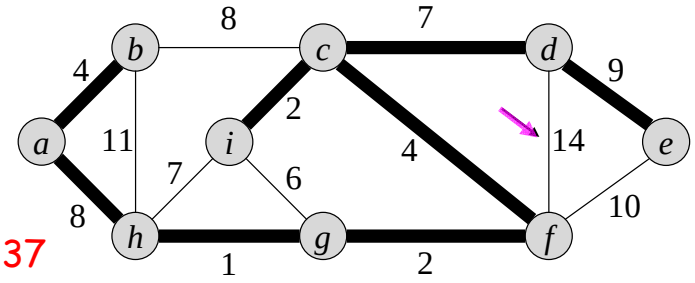


(g) 7th



(n) 13th

cost = 37



tree ➔ no cycles, connected

Time complexity:

Steps 1~3:	$O(V)$
Step 4:	$O(E \lg E)$ (sorting)
Steps 5~8:	$O(E\alpha(V))=O(E \lg E)$ (disjoint-set-forest in 21.3)

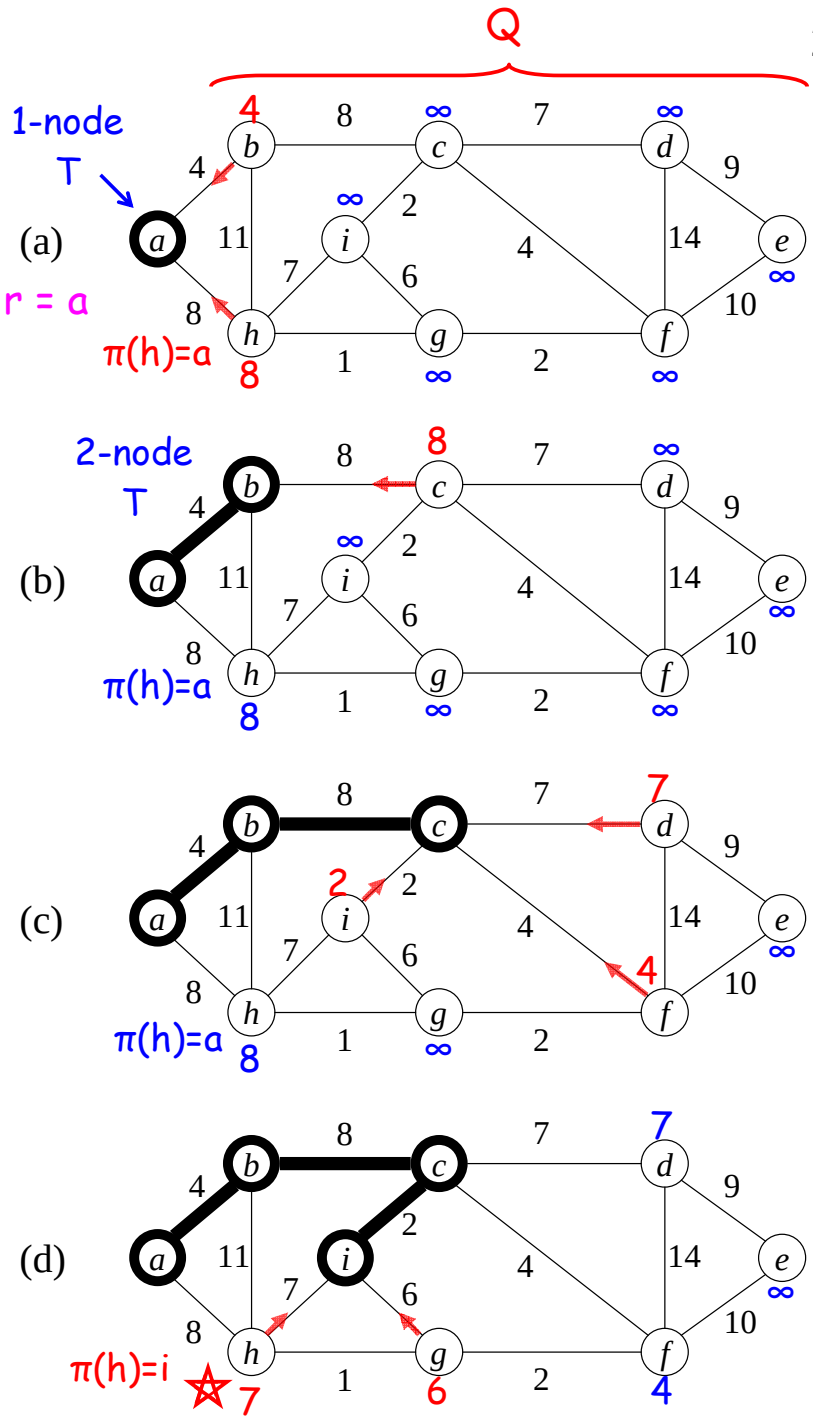
E×(set operation)

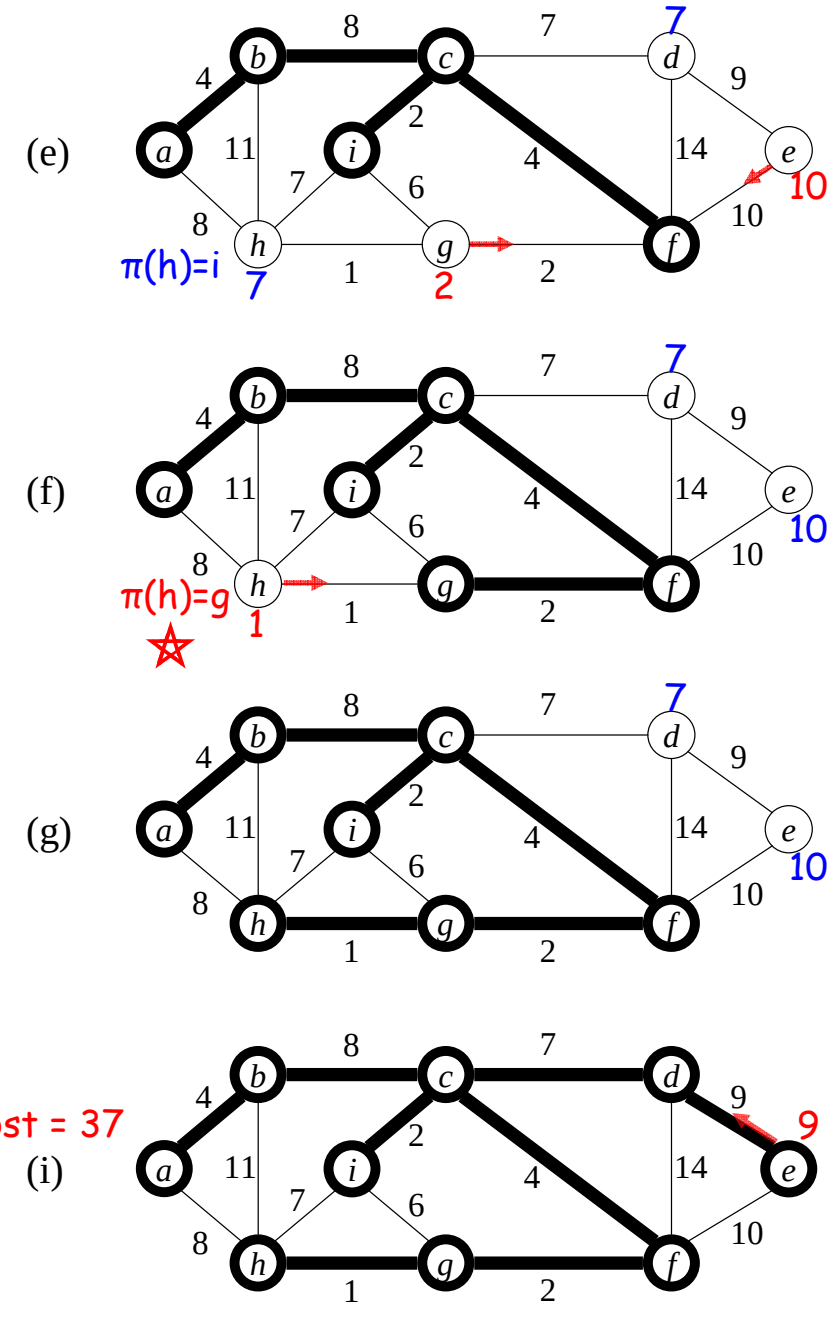
* α is the inverse Ackermann's function
* $\alpha(n) \le 4$ for for all practical cases
* $T(n)=O(E \lg E)$
* If all weights are bounded integers, $T(n)=O(E\alpha(V))$ ↳ integer sort

Prim's algorithm: vertices in A always form a single tree.

23-4a

```
MST-PRIM(G, w, r)
1  for each u ∈ V[G]                          和 current tree 的最小距離
2      do key[u] ← ∞                          Q: priority queue (vertices)
3         π[u] ← NIL                          parent
4  key[r] ← 0                                 build Q
5  Q ← V[G]                                   T 由 0 個 node 開始長
6  while Q ≠ ∅
7      do u ← EXTRACT-MIN(Q)                  V times
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)        decrease Key
                                              at most 2E times
```


Q
1-node
T
(a)
r = a
π(h)=a
2-node
T
(b)
π(h)=a
(c)
π(h)=a
(d)
π(h)=i


(e)
π(h)=i
(f)
π(h)=g
(g)
cost = 37
(i)

23-7x

Time complexity:

(a) Implement priority queue Q as an array

Steps 1~5: $O(V)$ (Build Q)
Step 7: $O(V^2)$ (V times Extract-Min)
Steps 8~11: $O(E)$ ($2E$ times Decrease-key)
Total: $O(V^2 + E) = O(V^2)$ (for dense G)

(b) Implement priority queue Q as a binary heap

Steps 1~5: $O(V)$ (Build Q)
Step 7: $O(V \lg V)$ (V times *Extract-Min*)
Steps 8~11: $O(E \lg V)$ ($2E$ times *Decrease-Key*)
Total: $O(E \lg V)$ (for sparse G)

(c) Implement Q as a Fibonacci heap

Steps 1~5: $O(V)$ (Build Q)
Step 7: $O(V \lg V)$ (V times *Extract-Min*)
Steps 8~11: $O(E)$ ($2E$ times *Decrease-Key*)
Total: $O(E + V \lg V)$ (for sparse G)

(Note $E \geq V - 1$)

Homework: Ex. 23.2-2, 23.2-4, 23.2-5, Prob. 23-1, 23-3.