Minimum Spanning Trees

m ≥ n - 1

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

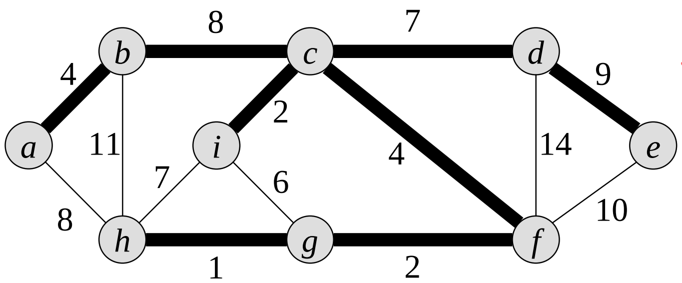
Output: A minimum spanning tree of G

23-1x

{n – 1 edges, no cycles

{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.

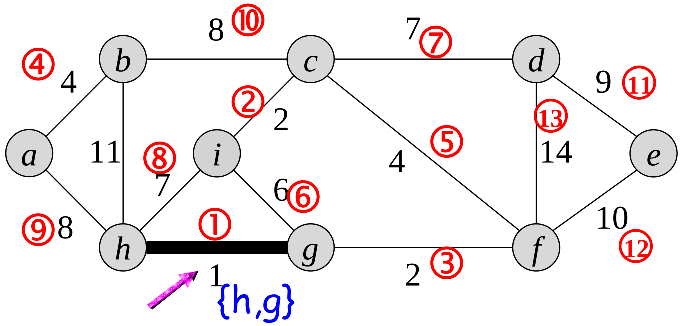
23.2 Kruskal's algorithm: smallest weighted first

(no cycle)

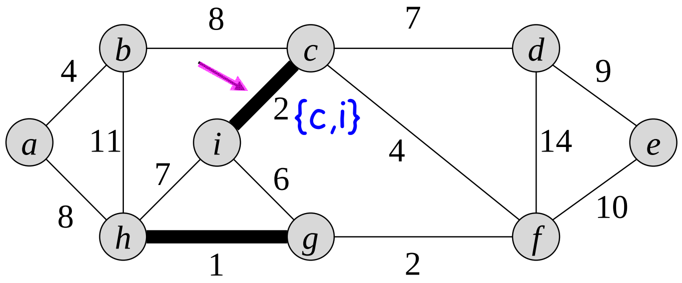
23-1x

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

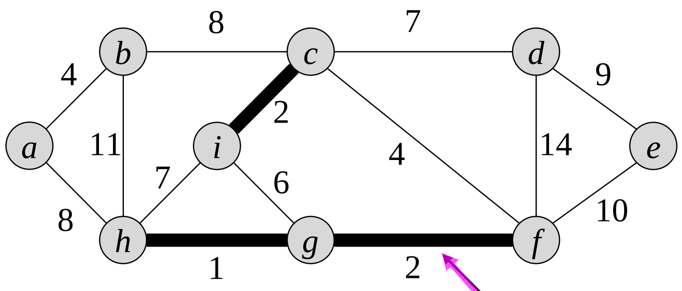

(a) 1st



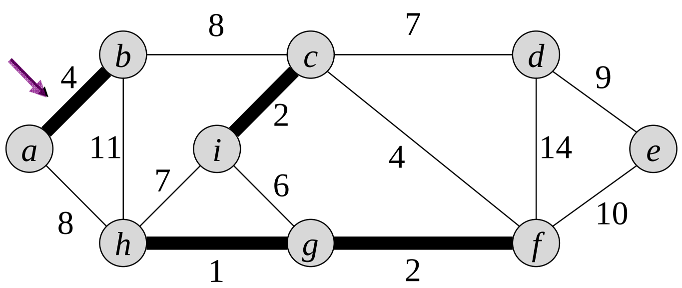
(b) 2nd



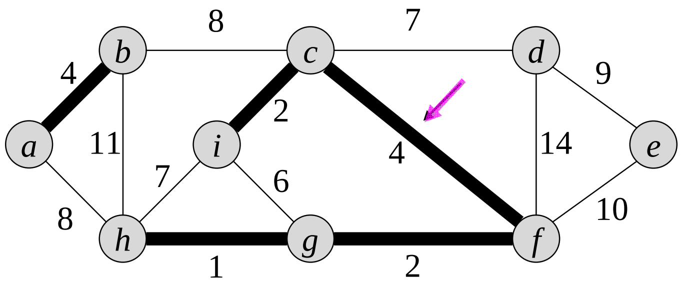
(c) 3rd



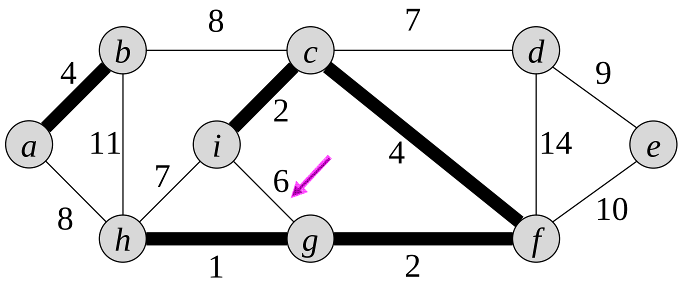
(d) 4th



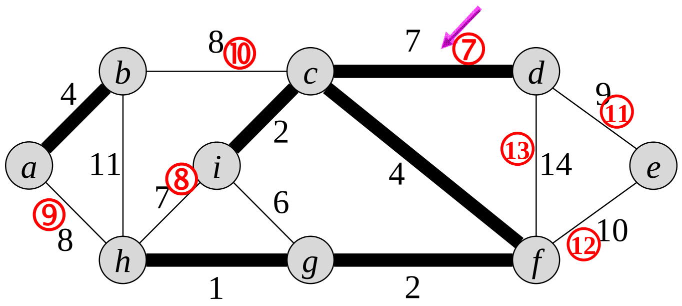
(e)
5th



☆(f)
6th

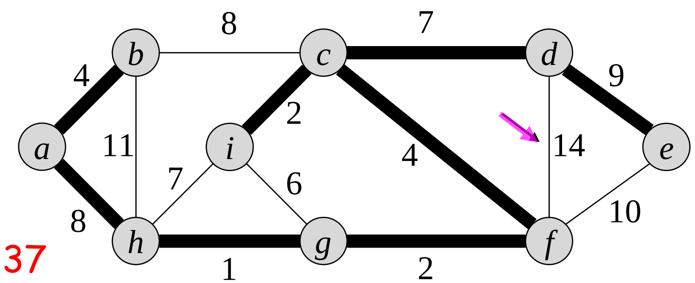


(g)
7th



⋮

☆(n)
13th
cost = 37



tree ➔ no cycles, connected

Time complexity:

Steps 1~3: $O(V)$

Step 4: $O(E\lg E)$ (sorting) E×(set operation)

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

* α is the inverse Ackermann's function
* $\alpha(n) \leq 4$ 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

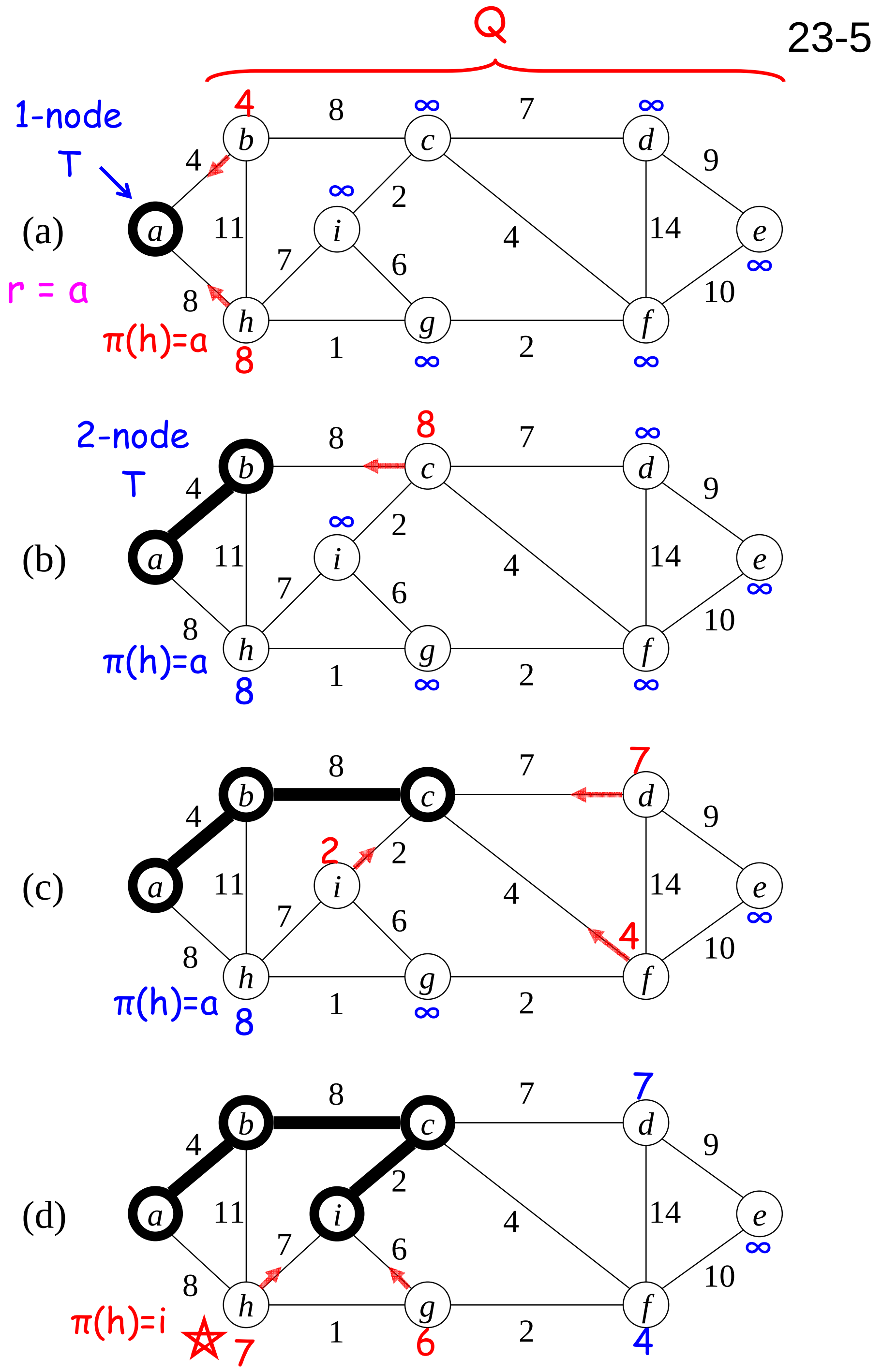
MST-PRIM(G, w, r) 和 current tree 的最小距離

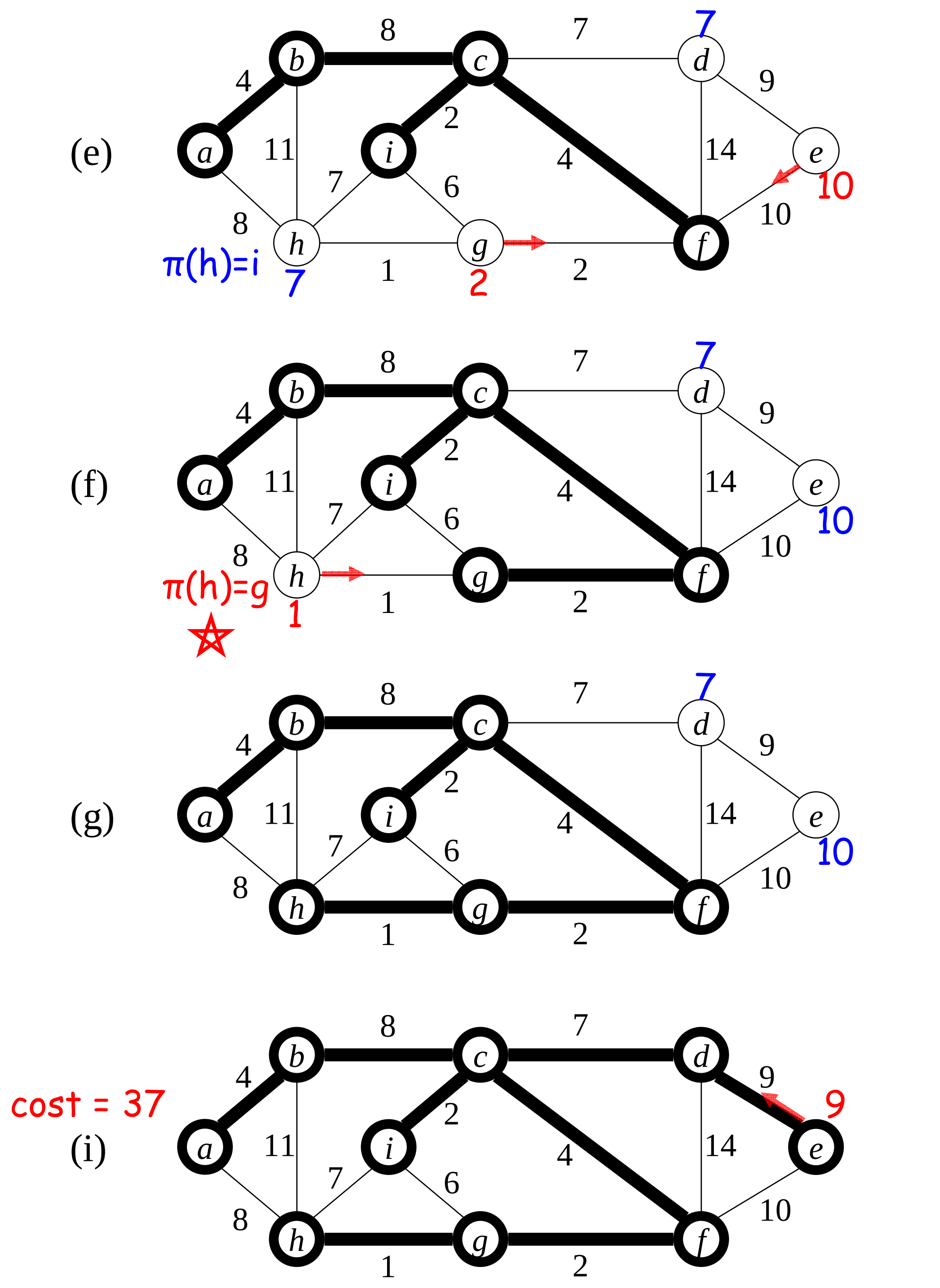
```
1   for each u ∈ V[G]
2       do key[u] ← ∞
3          π[u] ← NIL          parent
4   key[r] ← 0
5   Q ← V[G]                   build Q
6   while Q ≠ Ø
7       do u ← EXTRACT-MIN(Q)      V times
8          for each v ∈ Adj[u]
9              do if v ∈ Q and w(u, v) < key[v]
10                   then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

Q: priority queue (vertices)

T 由 0 個 node 開始長

at most 2E times


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


(e)
π(h)=i
(f)
π(h)=g
(g)
cost = 37
(i)
a
b
c
d
e
f
g
h
i

23-7x

Time complexity:

unsorted unsorted

(a) Implement priority queue Q as an array

solution to Ex. 23.2-2

Steps 1~5: $O(V)$ (Build Q) $O(V)$

Step 7: $O(V^2)$ (V times Extract-Min) $O(1)$

Steps 8~11: $O(E)$ ($2E$ times Decrease-key)

Total: $O(V^2 + E) = O(V^2)$ (for dense G) ~ simple

$(E \approx V^2)$

(b) Implement priority queue Q as a binary heap

Steps 1~5: $O(V)$ (Build Q) $O(\lg V)$

Step 7: $O(V \lg V)$ (V times *Extract-Min*)

Steps 8~11: $O(E \lg V)$ ($2E$ times *Decrease-Key*) $O(\lg V)$

Total: $O(E \lg V)$ (for sparse G)

$(E \ll V^2)$

(c) Implement Q as a Fibonacci heap

Steps 1~5: $O(V)$ (Build Q) $O(\lg V)$

Step 7: $O(V \lg V)$ (V times *Extract-Min*)

Steps 8~11: $O(E)$ ($2E$ times *Decrease-Key*) $O(1)$

Total: $O(E + V \lg V)$ (for sparse G)

(Note $E \geq V - 1$) $(E \ll V^2)$

☆

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