Graph Algorithms

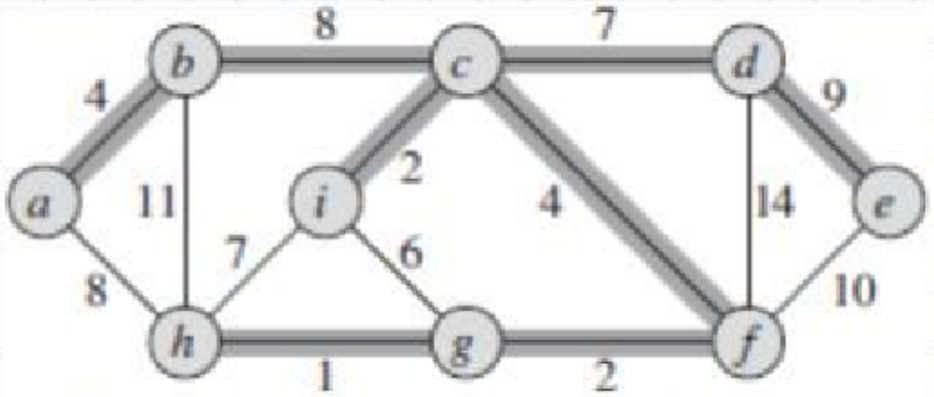
CS3104

Dr. Samit Biswas, *Assistant Professor,*
Department of Computer Sc. and Technology,
Indian Institute of Engineering Science and Technology, Shibpur

Email: samit@cs.iiests.ac.in

- **Minimum Spanning Tree**
  - Given a connected and undirected graph, a *spanning tree* of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees.
  - A ***minimum spanning tree (MST)*** or ***minimum weight spanning tree*** for a weighted, connected, undirected graph is a spanning tree with a weight less than or equal to the weight of every other spanning tree.
  - The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.
- ***How many edges does a minimum spanning tree has?***
  - A minimum spanning tree has (V – 1) edges where V is the number of vertices in the given graph.

Example



- A minimum spanning tree for a connected graph. The weights on edges are shown, and the edges in a minimum spanning tree are shaded. The total weight of the tree shown is **37**

- **Possible applications of MST**
  - Find the least expensive way to connect a set of cities, terminals, computers, etc.

Growing a MST - Generic Approach

- Grow a set A of edges (initially empty)
- Incrementally add edges to A such that they would belong to a MST
- **Idea: add only "safe" edges**
  - An edge (u, v) is **safe** for A if and only if $A \cup \{(u, v)\}$ is also a subset of **some** MST

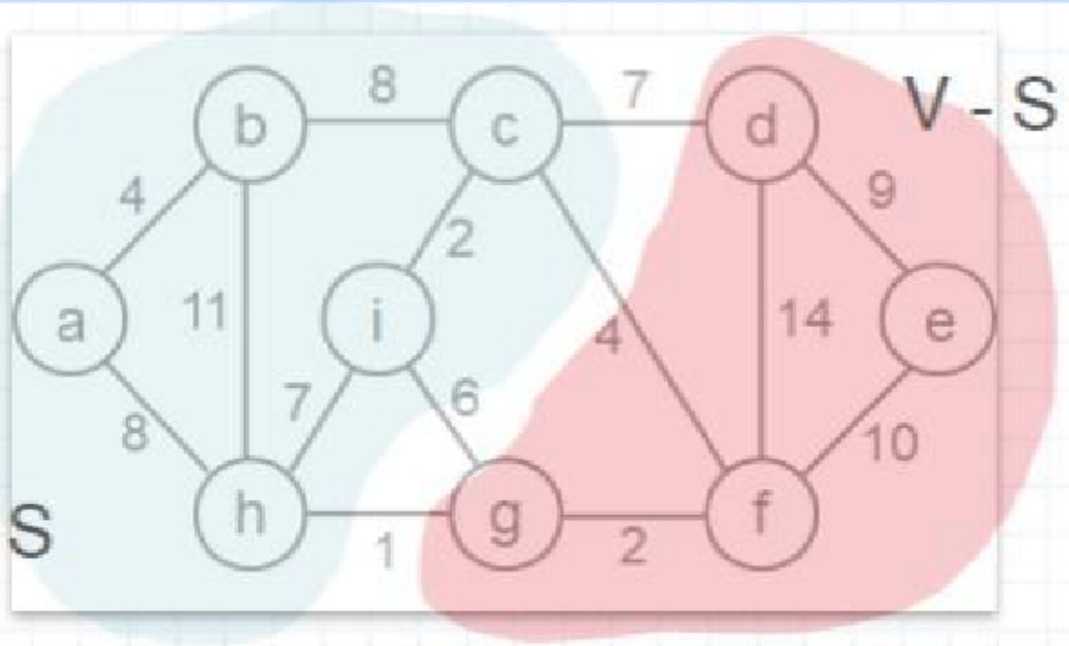
Growing a MST - Algorithm

1. $A \leftarrow \phi$
2. **while** A is not a spanning tree
3. **do** find an edge (u, v) that is safe for A
4. $A \leftarrow A \cup \{(u, v)\}$
5. **return** A

- How do we find safe edges ?

Finding Safe Edges

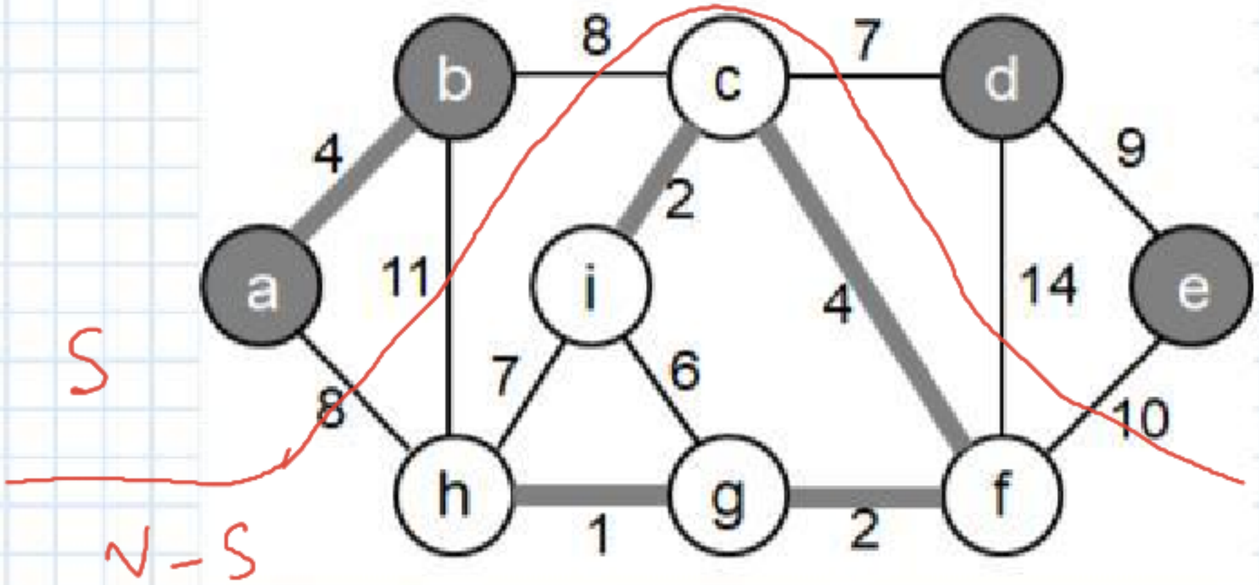
- Let's look at edge (h, g)
  - Is it safe for A initially?



- Later on:
  - Let S ⊂ V be any set of vertices that includes h but not g (so that g is in V - S)
  - In any MST, there has to be one edge (at least) that connects S with V - S
  - Why not choose the edge with **minimum weight** (h,g)?

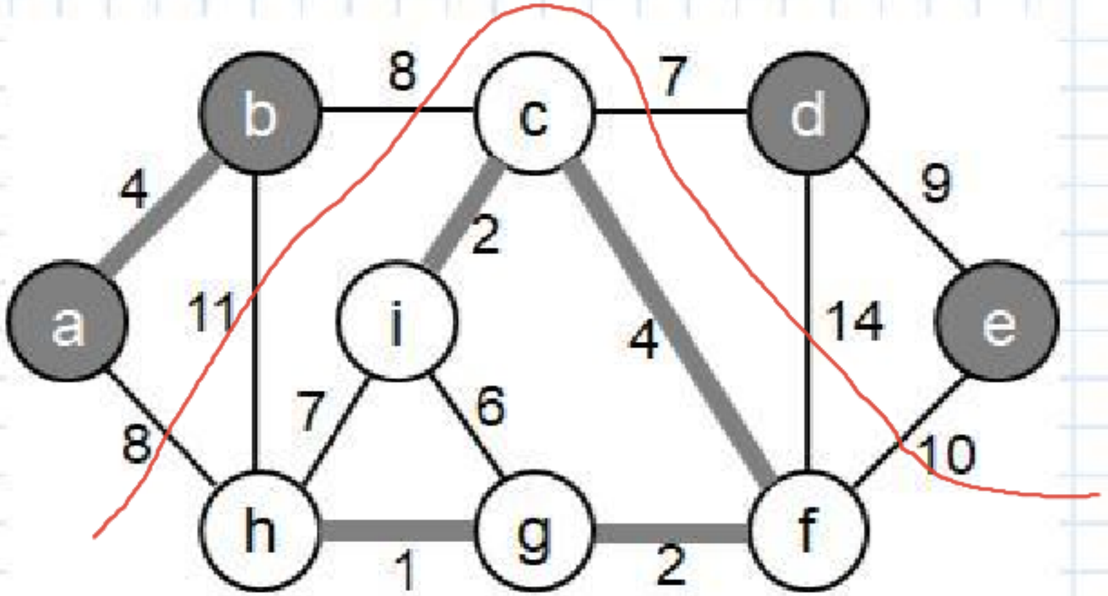
Definitions

- A **cut** (S, V - S) is a partition of vertices into disjoint sets S and V - S
- An edge **crosses** the cut (S, V - S) if one endpoint is in S and the other in V – S



Definitions

- A cut **respects** a set A of edges ⇔ no edge in A crosses the cut
- An edge is a **light edge** crossing a cut ⇔ its weight is minimum over all edges crossing the cut
  - Note that for a given cut, there can be > 1 light edges crossing it

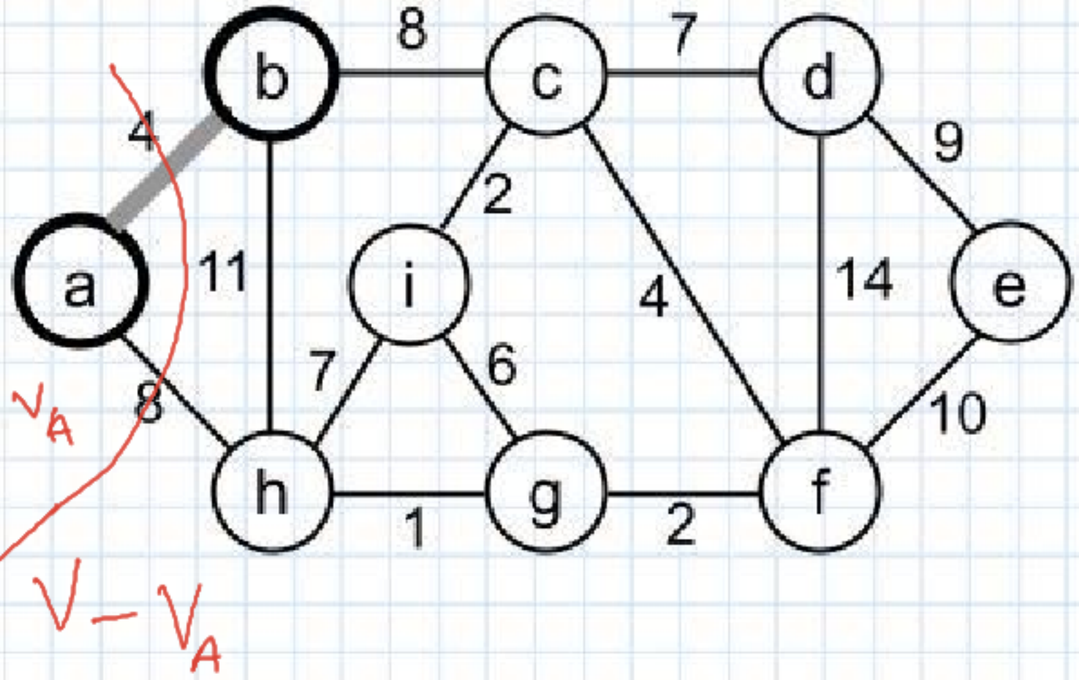


Algorithms for finding Minimum Spanning Tree

- Prim's algorithm
- Kruskal's algorithm

Prim's algorithm: finding Minimum Spanning Tree

- The edges in set A always form a single tree
- Starts from an arbitrary "root": $V_A = \{a\}$
- At each step:
  - Find a light edge crossing $(V_A, V - V_A)$
  - Add this edge to A
  - Repeat until the tree spans all vertices

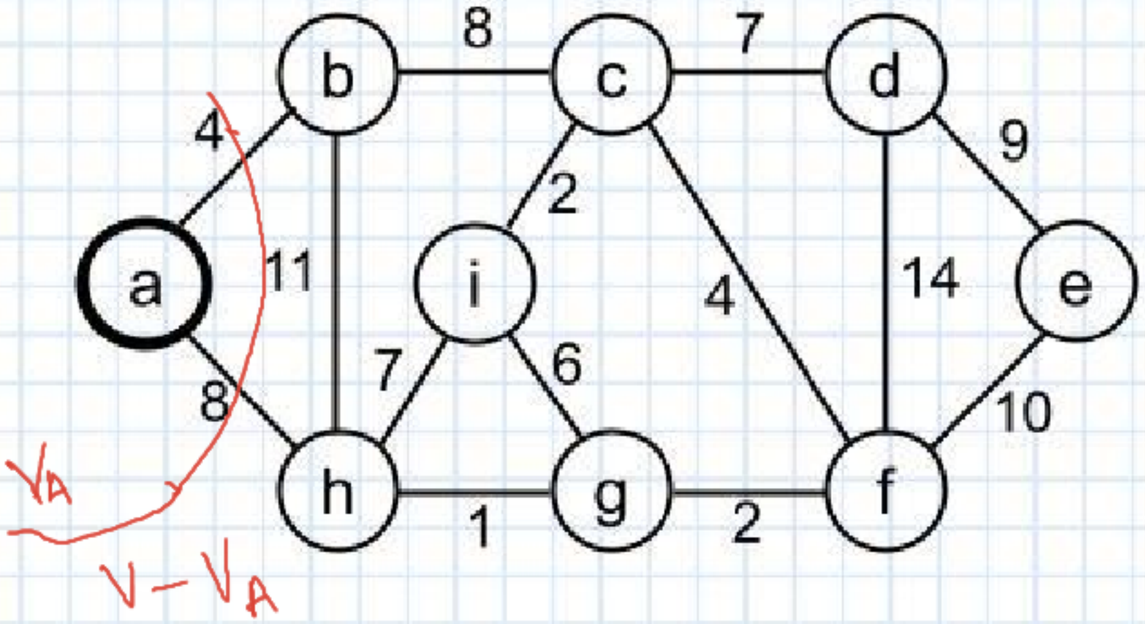


How to Find the light edge quickly ?

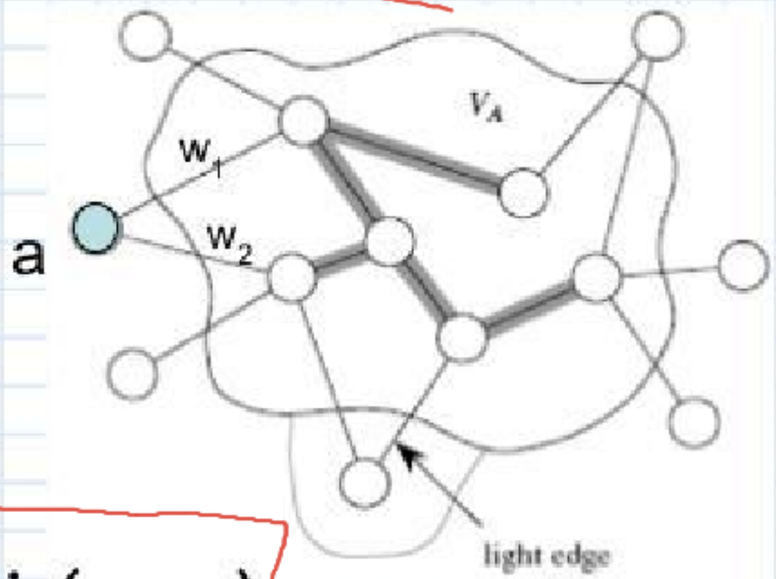
- Use a priority queue Q:
  - Contains vertices not yet included in the tree, i.e., $(V - V_A)$

  $V_A = \{a\}$, $Q = \{b, c, d, e, f, g, h, i\}$
- We associate a key with each vertex v:

  key[v] = minimum weight of any edge (u, v) connecting v to V_A



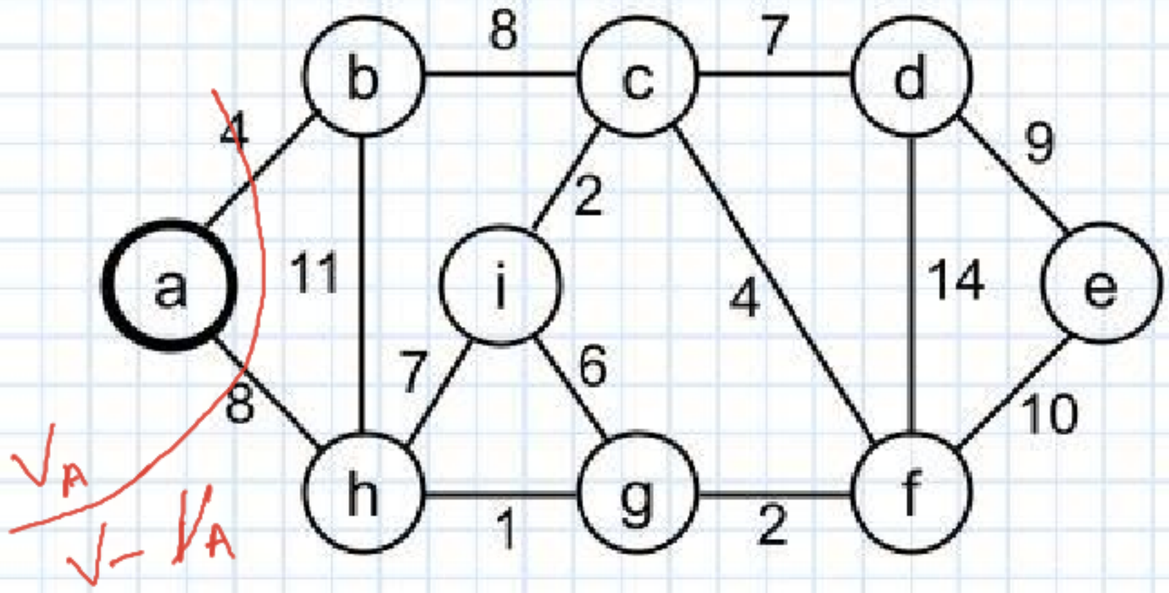
k[b] = 4



Key[a] = $min(w_1, w_2)$

How to Find the light edge quickly ? (Contd ...)

- After adding a new node to V_A we update the weights of all the nodes adjacent to it
  - e.g., after adding a to the tree, k[b]=4 and k[h]=8
- Key of v is ∞ if v is not adjacent to any vertices in V_A

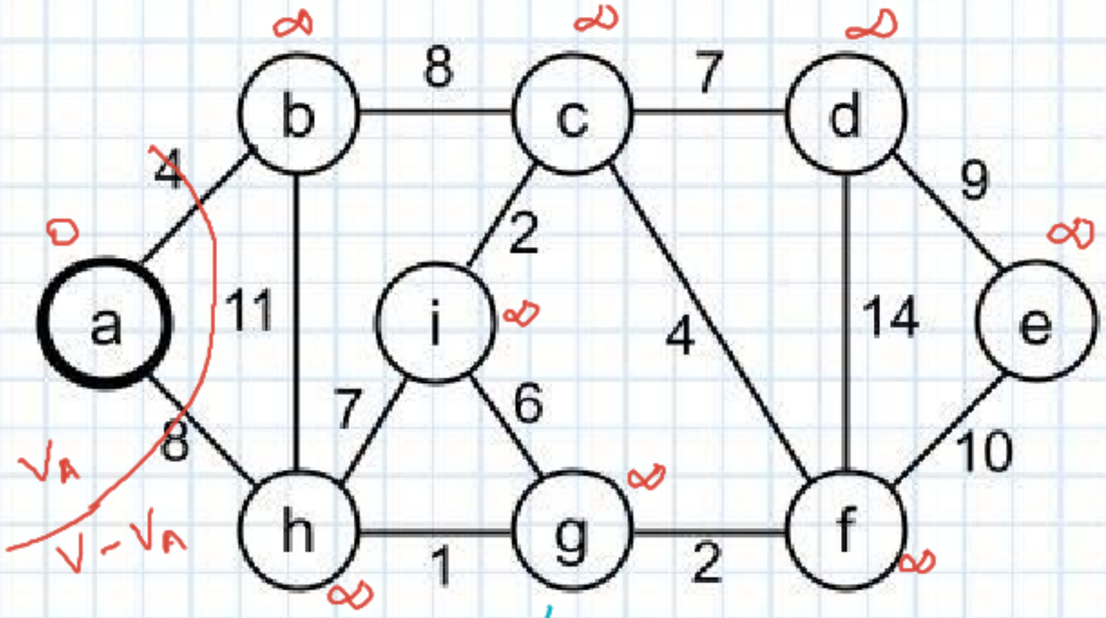


$V_A = \{a\}$

Key [b] = 4

Key [h] = 8

Example



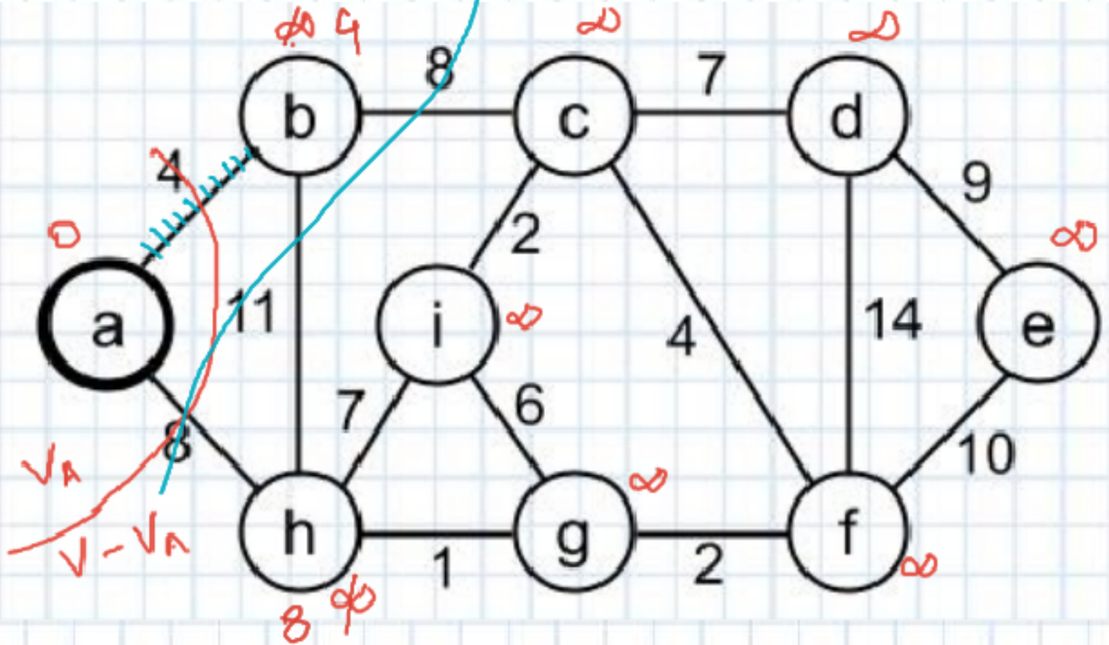
$Q = \{a, b, c, d, e, f, g, h, i\}$

$V_A = \emptyset$

Extract-min(Q) $\Rightarrow$ a

key[b] = 4 $\quad \pi[b] = a$

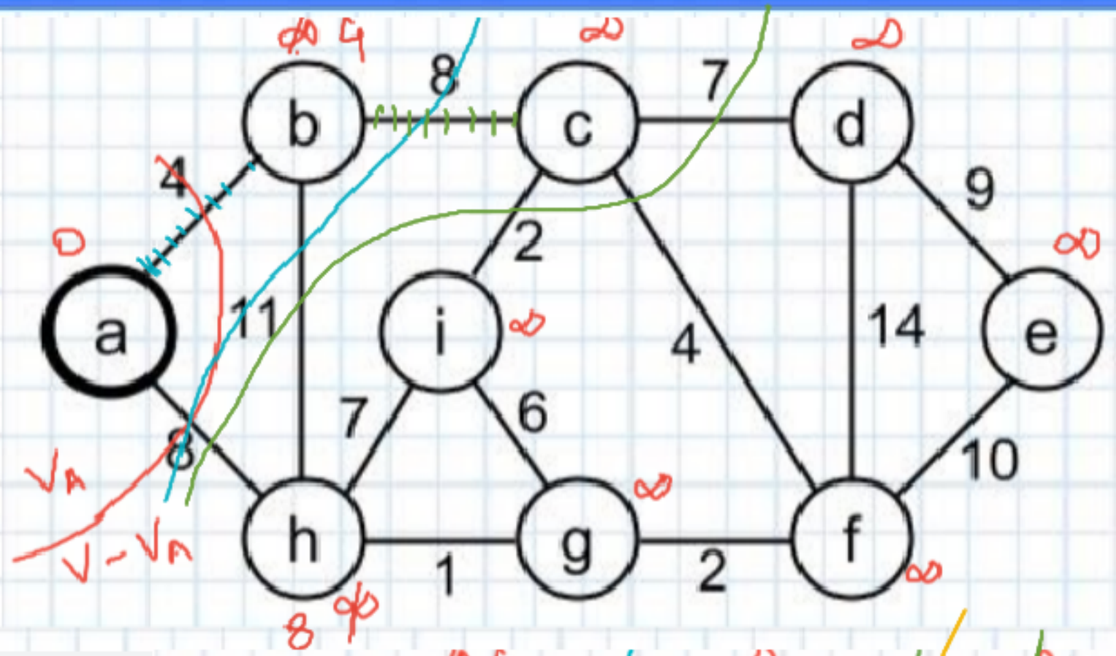
key[h] = 8 $\quad \pi[h] = a$

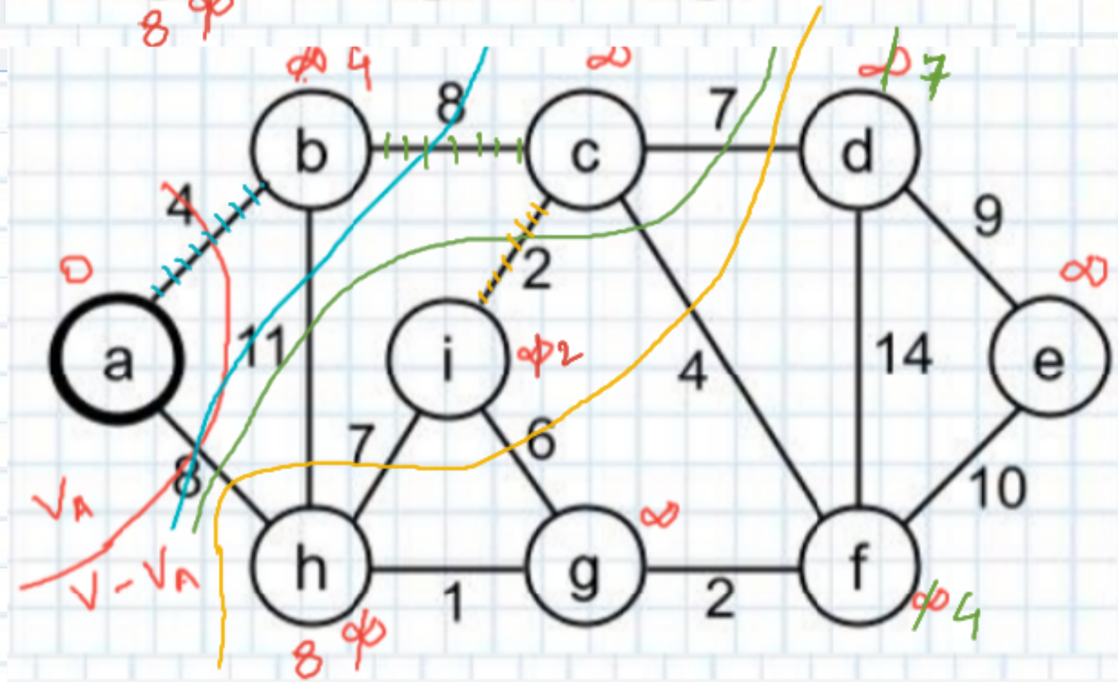


$Q = \{b, c, d, e, f, g, h, i\}$, $V_A = \{a\}$

Extract-min(Q) $\Rightarrow$ b

Example





Key[c] = 8 $\quad$ $\pi[c] = b$

Key[h] = 8 $\quad$ $\pi[h] = a$ unchanged

$Q = \{c, d, e, f, g, h, i\}$, $V_A = \{a, b\}$

Extract-min(Q) = c

Key[d] = 7 $\quad$ $\pi[d] = c$

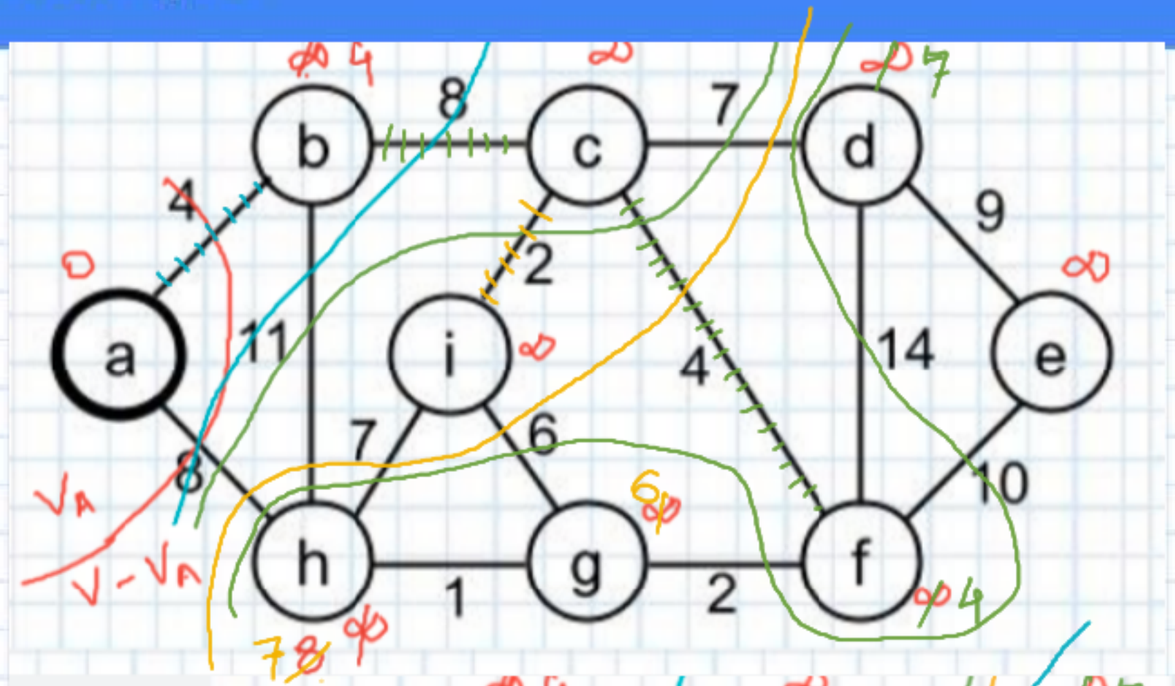
Key[f] = 4 $\quad$ $\pi = c$

Key[i] = 2 $\quad$ $\pi = c$

$Q = \{d, e, f, g, h, i\}$, $V_A = \{a, b, c\}$

Extract-Min(Q) $\Rightarrow$ i

Example



Key[h] = 7 $\pi[h] = i$

Key[g] = 6 $\pi[g] = i$

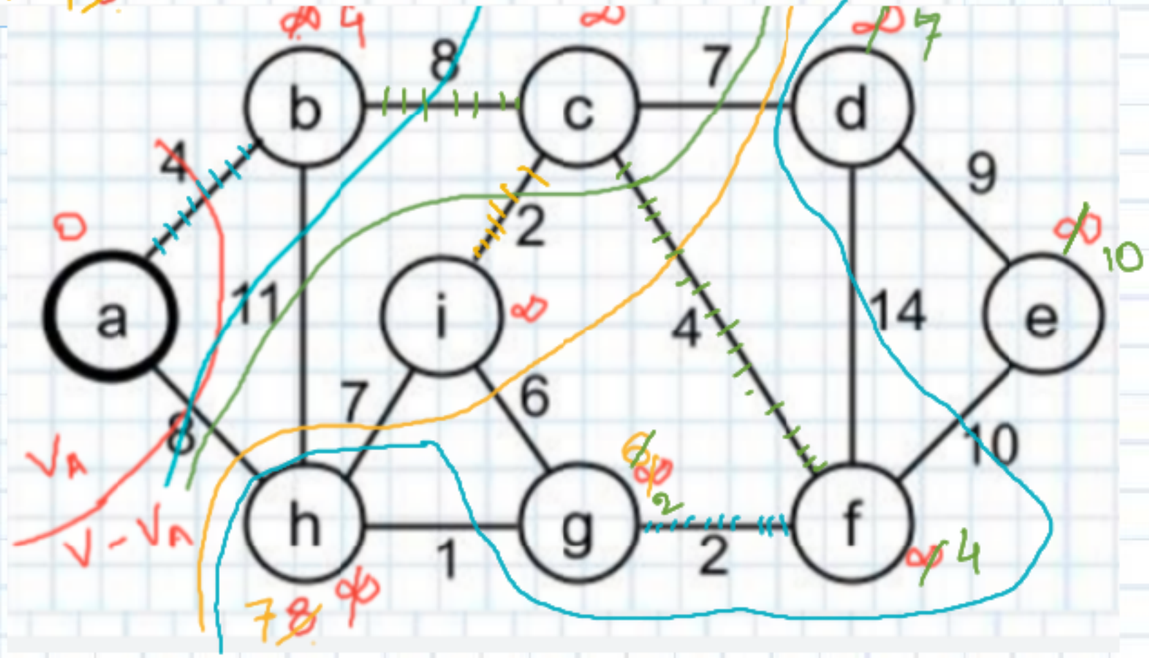
$Q = \{d^{7}, e^{\infty}, f^{4}, g^{6}, h^{7}\}$, $V_A = \{a, b, c, i\}$

Extract-MIN(Q) $\Rightarrow$ f

Key[g] = 2 $\pi[g] = f$

Key[d] = 7 $\pi[d] = c$, unchanged

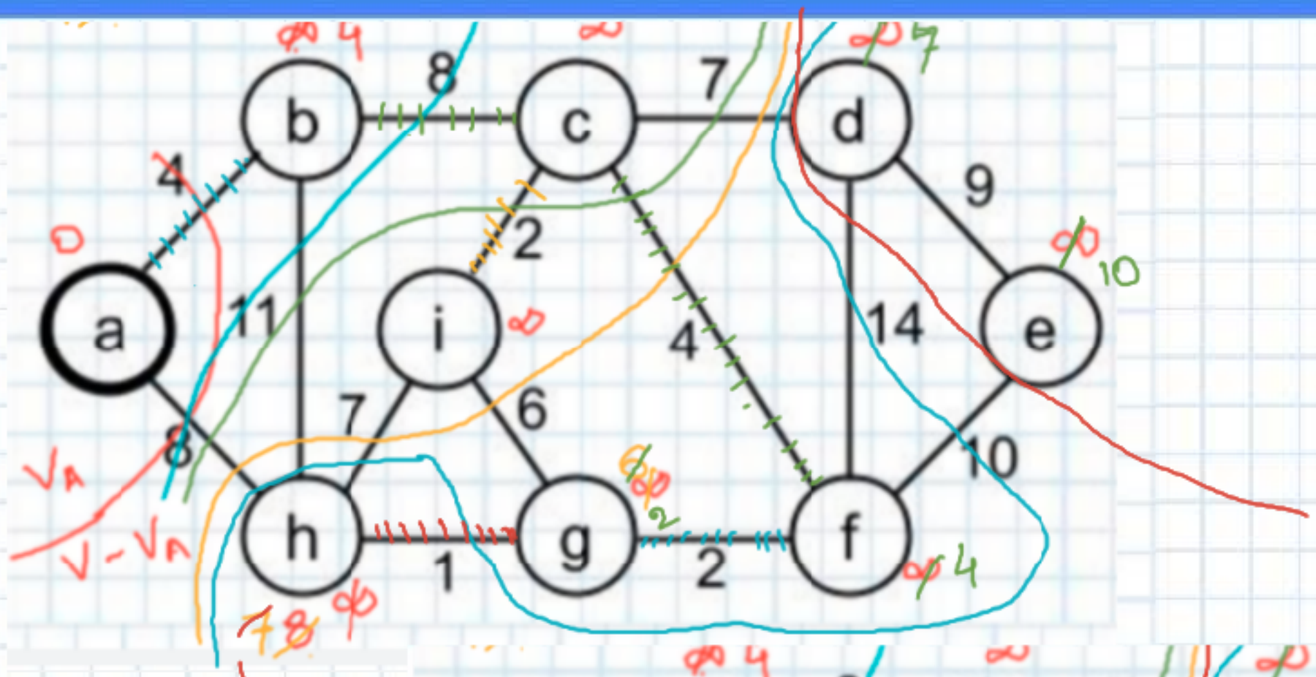
Key[e] = 10 $\pi[e] = f$



$Q = \{d^{7}, e^{10}, g^{2}, h^{8}\}$, $V_A = \{a, b, c, i, f\}$

Extract-MIN(Q) $\Rightarrow$ g

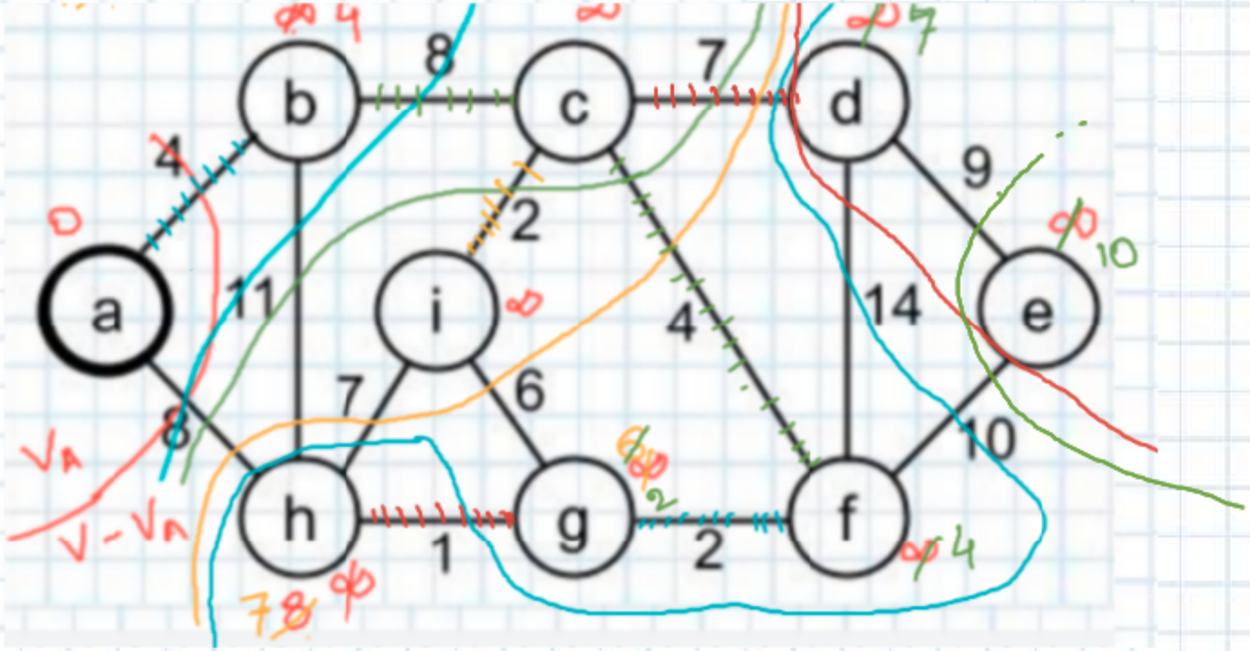
Example



key[h] = 1 $\pi[h] = g$

$Q = \{d, e, h\}$. $V_A = \{a, b, c, i, f, g\}$

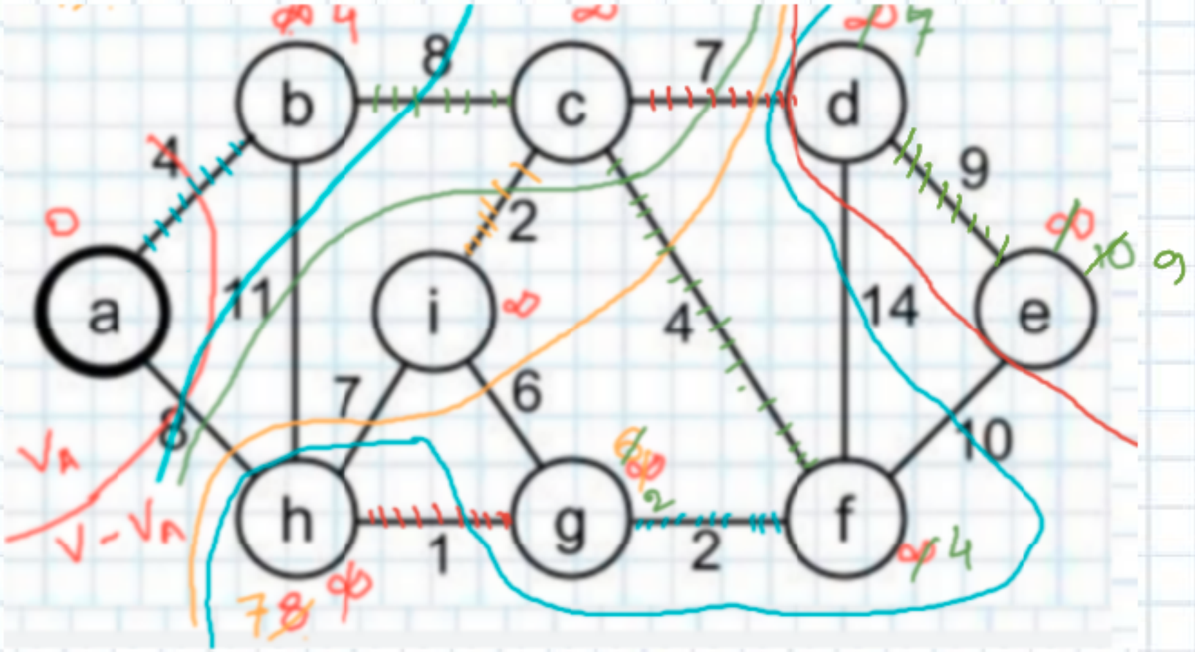
Extract-MIN(Q) $\Rightarrow$ h



$Q = \{d, e\}$, $V_A = \{a, b, c, i, f, g, h\}$

Extract $\Rightarrow$ d

Example



key[e] = 9 , π(e) = f

Q = { e } , V_A = { a, b, c, i, f, g, h, d }

Extract-MIN(Q) ⇒ e

Q = ∅ , V_A = { a, b, c, d, e, f, g, h, i }

Analysis of Prim's algorithm

```
Algorithm Prim(V, E, w, r)
1.  Q ← ∅
2.  for each u ∈ V
3.      do key[u] ← ∞
4.         π[u] ← NIL
5.         INSERT(Q, u)
6.  DECREASE-KEY(Q, r, 0)        ▶ key[r] ← 0
7.  while Q ≠ ∅
8.      do u ← EXTRACT-MIN(Q)
9.         for each v ∈ Adj[u]
10.           do if v ∈ Q and w(u, v) < key[v]
11.                 then π[v] ← u
12.                      DECREASE-KEY(Q, v, w(u, v))
```

O(V) if Q is implemented with min-heap.

Total Time : $O(V \lg V + E \lg V) = O(E \lg V)$

→ $O(\lg V)$

→ Executed $|V|$ number of times

Taken $O(\lg V)$

} $O(V \lg V)$

→ Executes $O(E)$ times

→ constant

$O(E \lg V)$

$O(\lg V)$