5CCS2FC2: Foundations of Computing II

Introduction to Graph Algorithms

Week 4

Dr Christopher Hampson

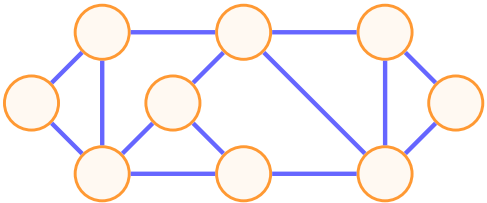
Department of Informatics

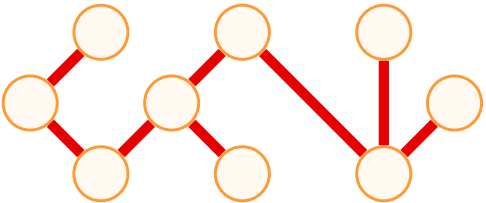
King's College London

Minimum Spanning Trees

Spanning Tree

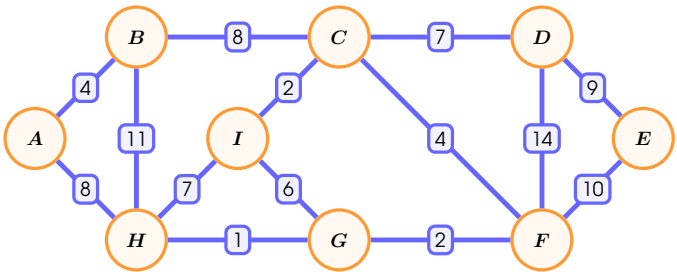
- A **spanning tree** for a weighted graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{w})$ is a *tree* $\boldsymbol{T} = (\boldsymbol{V}, \boldsymbol{E'})$ in which each vertex is connected and $\boldsymbol{E'} \subseteq \boldsymbol{E}$,





Minimum Spanning Tree (MST)

- A **minimum spanning tree** is a spanning tree whose *weight* is minimal out of all possible spanning trees



$\sum$ weights $= 37$

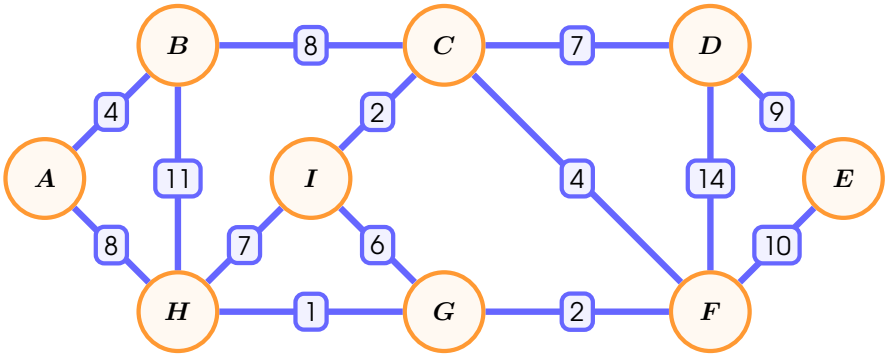
Kruskal's Minimum Spanning Tree Algorithm

Kruskal's Spanning Tree Algorithm

KRUSKAL-MST (G) :

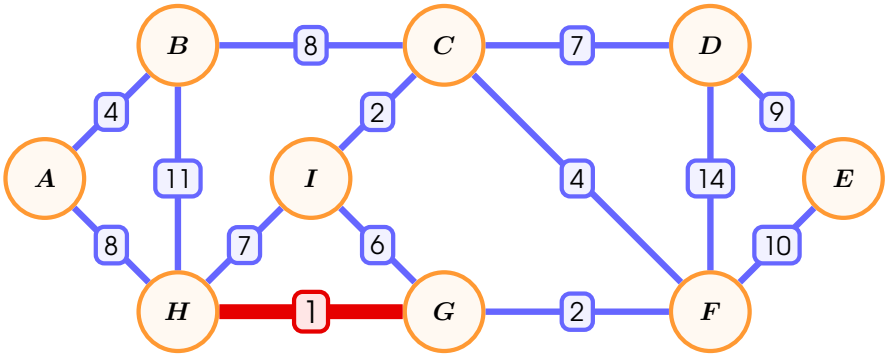
1. Sort the edge list E by weight (shortest first)
2. Initialise the set $T = \emptyset$
3. **While** sorted list E is not-empty:
    4. Dequeue shortest edge (u, v) from E
    5. **If** If $T \cup \{(u, v)\}$ is acyclic:
        6. Update $T \leftarrow T \cup \{(u, v)\}$.
7. **Return** tree T

Kruskal's Spanning Tree Algorithm



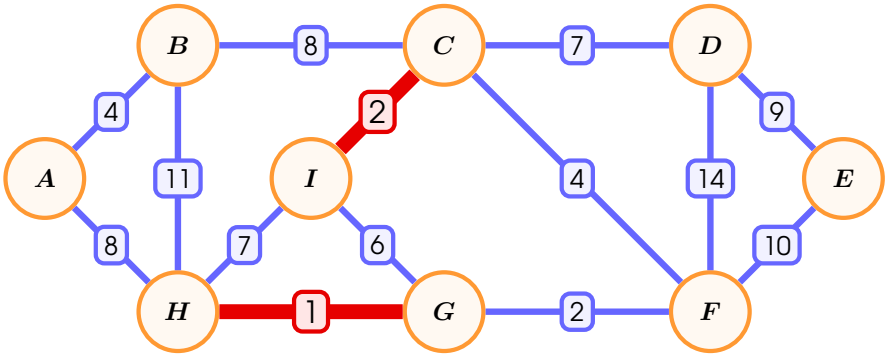
$\{A\}$ $\{B\}$ $\{C\}$ $\{D\}$ $\{E\}$ $\{F\}$ $\{G\}$ $\{H\}$ $\{I\}$

Kruskal's Spanning Tree Algorithm



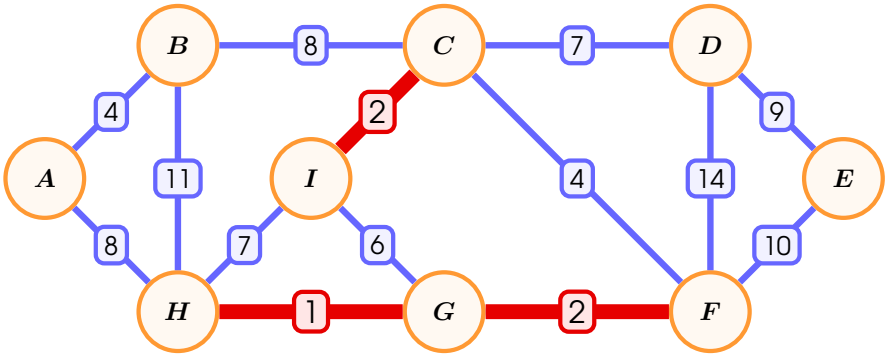
$\{A\}$ $\{B\}$ $\{C\}$ $\{D\}$ $\{E\}$ $\{F\}$ $\{G, H\}$ $\{I\}$

Kruskal's Spanning Tree Algorithm



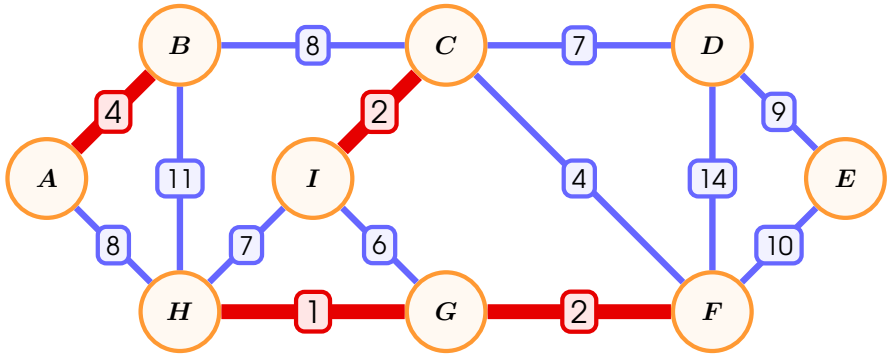
$\{A\}$ $\{B\}$ $\{D\}$ $\{E\}$ $\{F\}$ $\{G, H\}$ $\{C, I\}$

Kruskal's Spanning Tree Algorithm



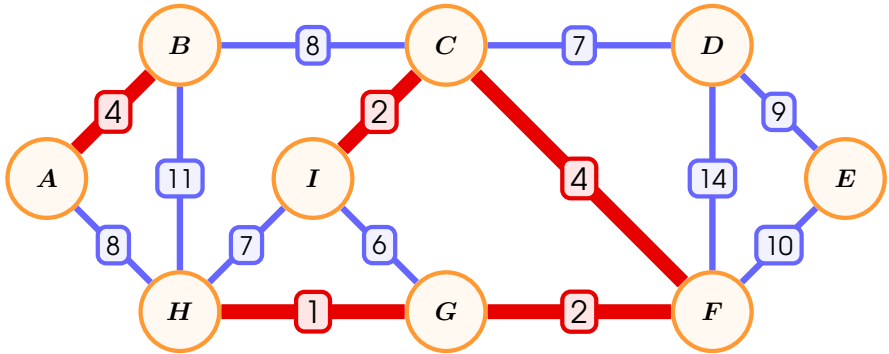
$\{A\}$ $\{B\}$ $\{D\}$ $\{E\}$ $\{F, G, H\}$ $\{C, I\}$

Kruskal's Spanning Tree Algorithm



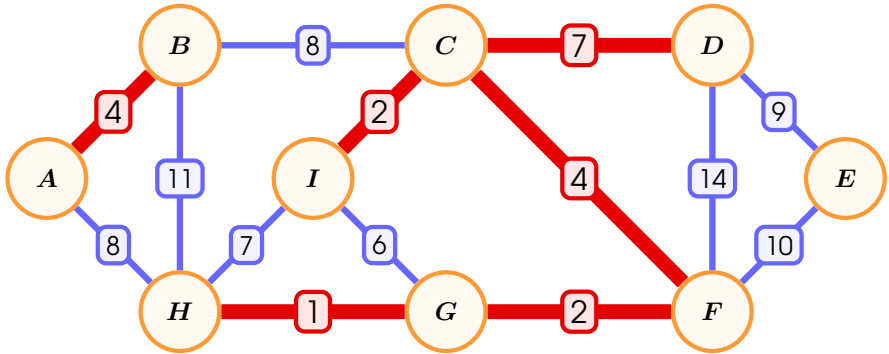
$\{A, B\}$ $\{D\}$ $\{E\}$ $\{F, G, H\}$ $\{C, I\}$

Kruskal's Spanning Tree Algorithm



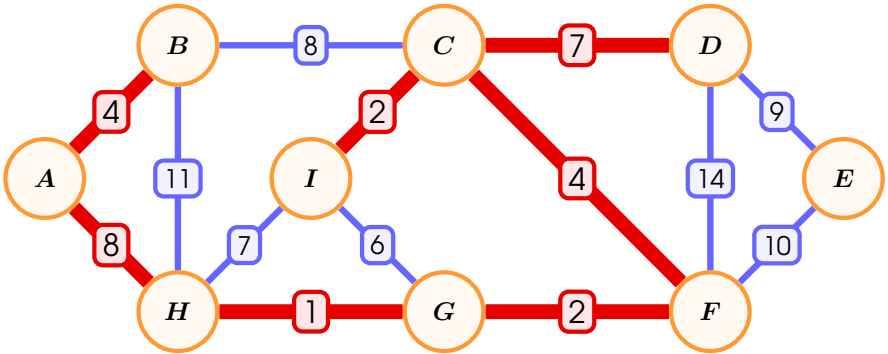
$\{A, B\}$ $\{D\}$ $\{E\}$ $\{F, G, H, C, I\}$

Kruskal's Spanning Tree Algorithm



$\{A, B\}$ $\{E\}$ $\{D, F, G, H, C, I\}$

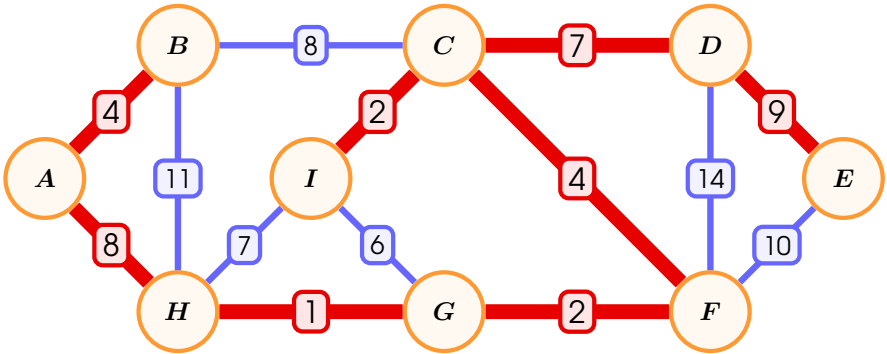
Kruskal's Spanning Tree Algorithm



$\{E\}$

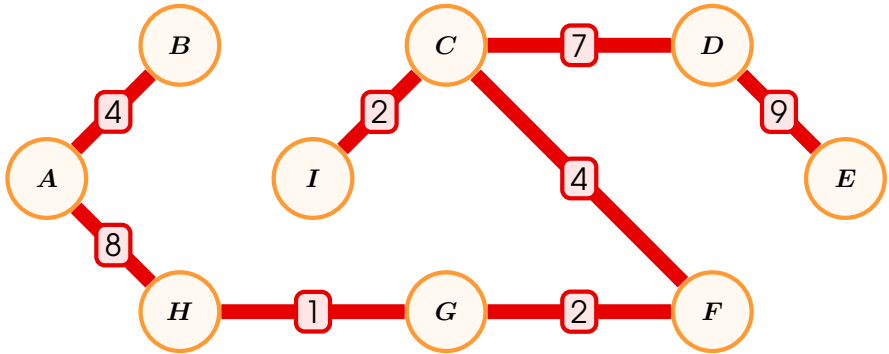
$\{A, B, F, G, H, C, I, D\}$

Kruskal's Spanning Tree Algorithm



$\{A, B, F, G, H, C, I, D, E\}$

Kruskal's Spanning Tree Algorithm



Kruskal's Spanning Tree Algorithm

Theorem The termination time for Kruskal's Algorithm is $O(|E| \log |E|)$.

Proof:

- For **Step 1** we need to sort E which can be done via **merge-sort** or some other efficient sorting algorithm:

time to sort E by weight $= O(|E| \log |E|)$

Kruskal's Spanning Tree Algorithm

Theorem The termination time for Kruskal's Algorithm is $O(|E| \log |E|)$.

Proof:

- **Steps 4 and 6** are **primative** and take $O(1)$ steps, for each iteration $(u, v) \in E$.

$$\text{total time for Steps 4 and 6} = \sum_{E} (O(1) + O(1)) = O(|E|)$$

Kruskal's Spanning Tree Algorithm

Theorem The termination time for Kruskal's Algorithm is $O(|E| \log |E|)$.

Proof:

- **Step 5** can be performed by checking whether u and v are already **connected**, since a second connection would create a cycle.

$$\text{total time to check connectedness} \;=\; O\left((|V| + |E|)\ \alpha(|V|)\right)$$

where α is a *very very!* slow growing function that we can forget about.

(if interested, see Cormen et al. Section 21.3 for more details)

Kruskal's Spanning Tree Algorithm

Theorem The termination time for Kruskal's Algorithm is $O(|E| \log |E|)$.

Proof:

- Hence, the total **worst-case termination time** is given by

$$T(V, E) = O(|E| \log |E|) + O(|E|) + O((|V| + |E|)\, \alpha(|V|))$$

Q.E.D.

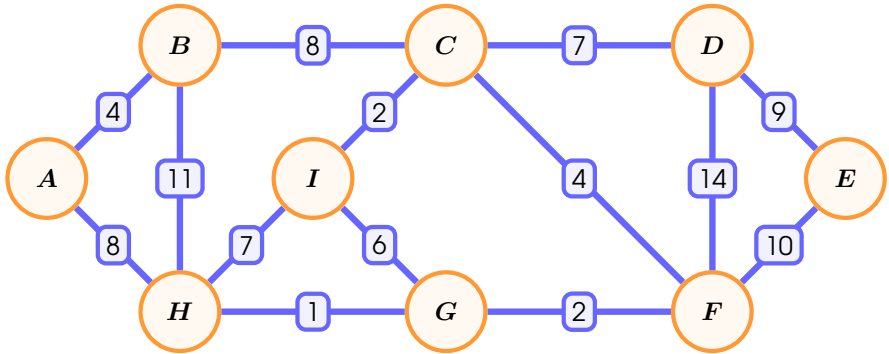
Prim's Minimum Spanning Tree Algorithm

Prim's Spanning Tree Algorithm

PRIM-MST (G) :

1. Select a root node $r \in V$ and all adjacent edges to a priority queue Q
2. Initialise the set $T = \emptyset$
3. **While** queue Q is not-empty:
    4. Dequeue shortest edge (u, v) from Q
    5. **If** $T \cup \{(u, v)\}$ is acyclic:
        6. Update $T \leftarrow T \cup \{(u, v)\}$.
        7. Add to Q any new edges adjacent to u or v.
8. **Return** tree T

Prim's Spanning Tree Algorithm



INSERT

(A, B)

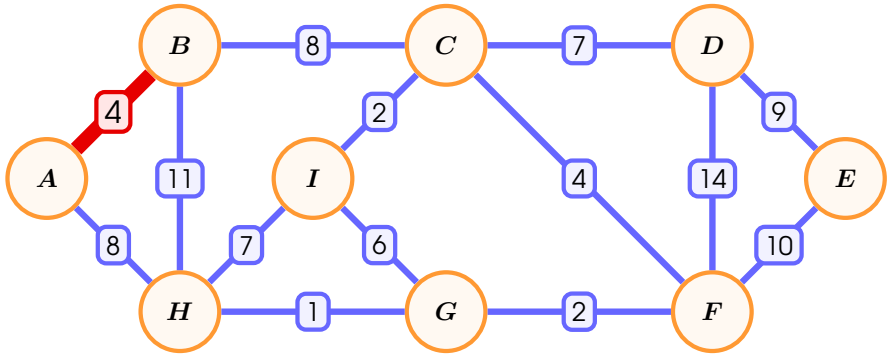
4

INSERT

(A, H)

8

Prim's Spanning Tree Algorithm



INSERT

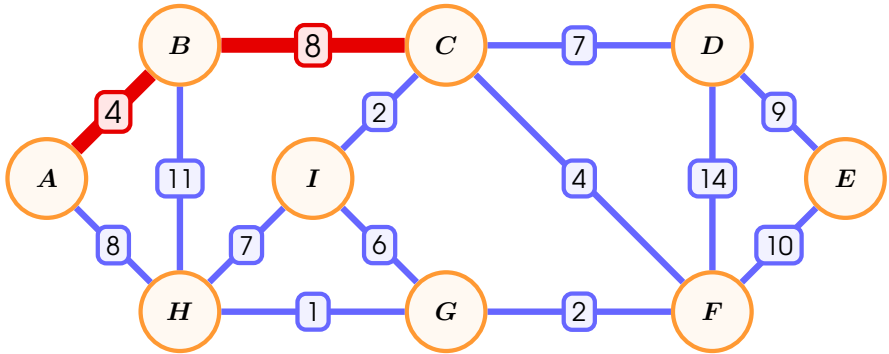
(B, C)
8

(A, H)
8

INSERT

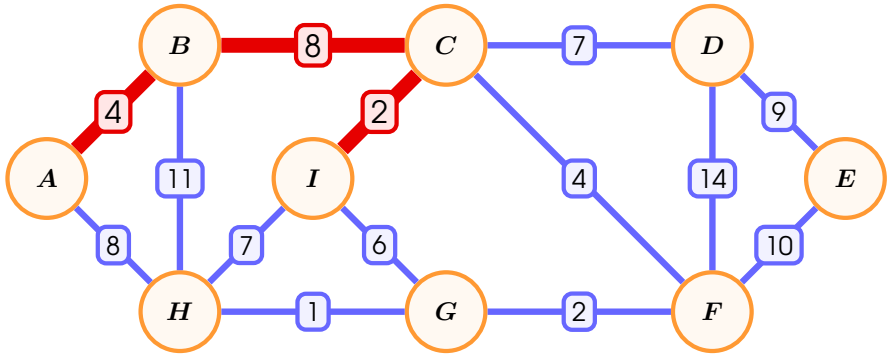
(B, H)
11

Prim's Spanning Tree Algorithm



INSERT	INSERT	INSERT		
(C, I)	(C, F)	(C, D)	(A, H)	(B, H)
2	4	7	8	11

Prim's Spanning Tree Algorithm



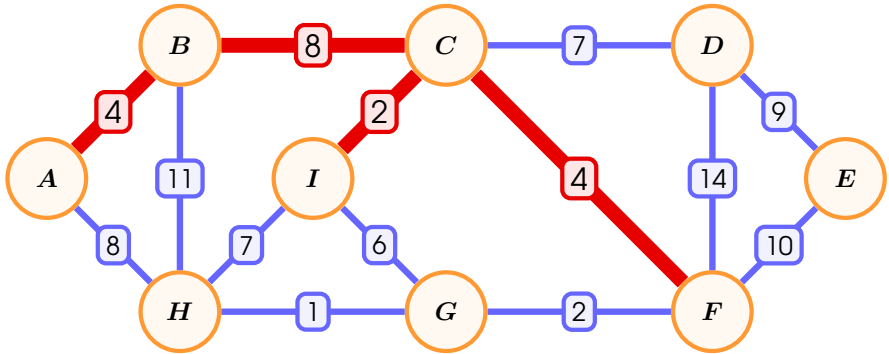
INSERT

(C, F)	(I, G)	(C, D)
4	6	7

INSERT

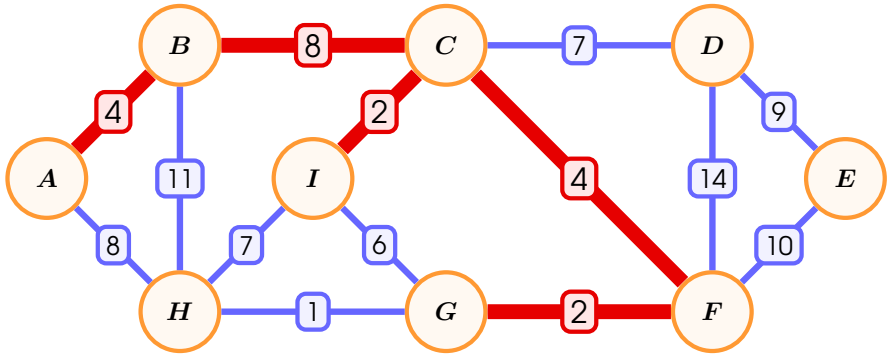
(I, H)	(A, H)	(B, H)
7	8	11

Prim's Spanning Tree Algorithm



INSERT					INSERT		INSERT
(F, G) 2	(I, G) 6	(C, D) 7	(I, H) 7	(A, H) 8	(F, E) 10	(B, H) 11	(F, D) 14

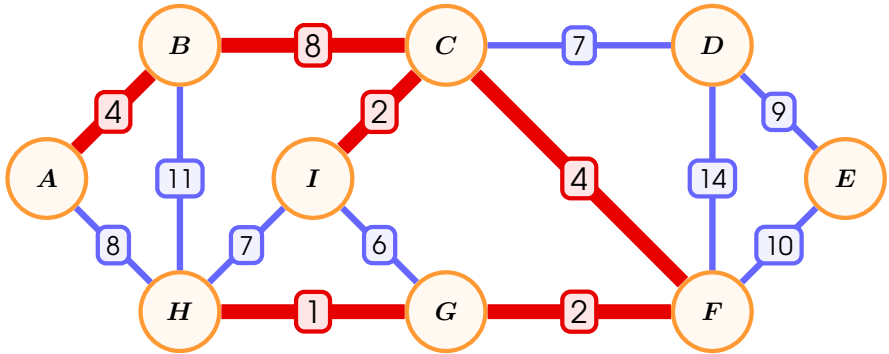
Prim's Spanning Tree Algorithm



INSERT

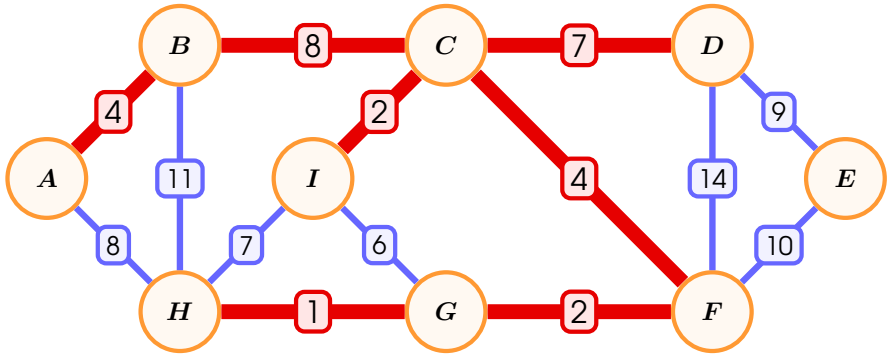
(G, H)	(I, G)	(C, D)	(I, H)	(A, H)	(F, E)	(B, H)	(F, D)
1	6	7	7	8	10	11	14

Prim's Spanning Tree Algorithm



$(I, \ldots)$	(C, D)	(I, H)	(A, H)	(F, E)	(B, H)	(F, D)
6	7	7	8	10	11	14

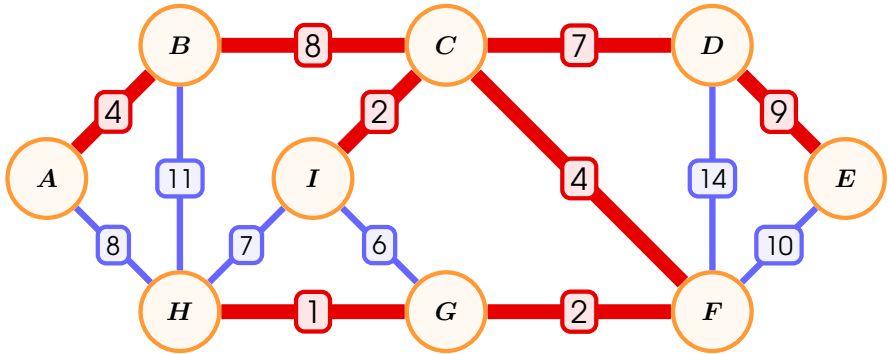
Prim's Spanning Tree Algorithm



INSERT

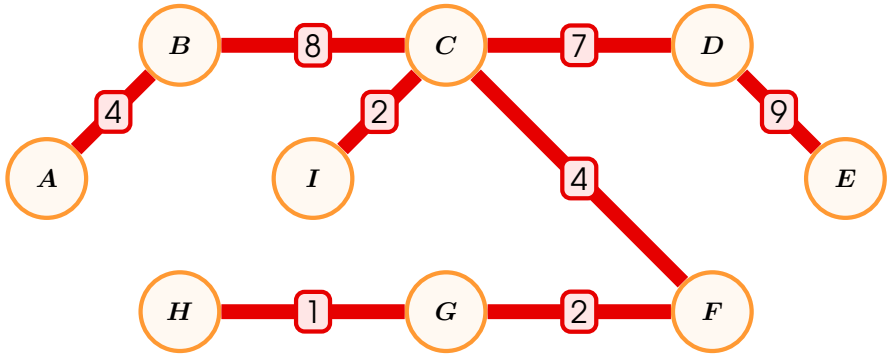
(I, H)	(A, B)	(D, E)	(F, E)	(B, H)	(F, D)
~~7~~	~~8~~	9	10	11	14

Prim's Spanning Tree Algorithm



(F, E)	(B, H)	(F, D)
10	11	14

Prim's Spanning Tree Algorithm



Prim's Spanning Tree Algorithm

Theorem The termination time for Prims's Algorithm is $O(|E|+|V|\log|V|)$.

Proof:

- Subject to the data structure used for maintaining the **priority queue**.
- Utilising a data structure called a **Fibonacci Heap**, we can obtain the above complexity bound.

(if interested, see Cormen et al. Chapter 19 for more details)

Q.E.D.

End of Slides!