Data Structures – LECTURE 13

Minumum spanning trees

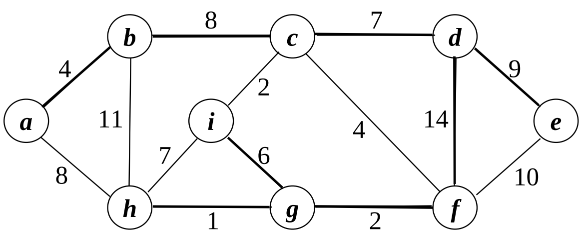
- Motivation
- Properties of minimum spanning trees
- Kruskal's algorithm
- Prim's algorithm

Chapter 23 in the textbook (pp 561—579).

Motivation

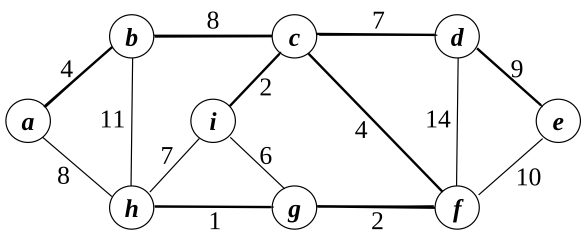
- Given a set of nodes and possible connections with weights between them, find the subset of connections that connects all the nodes *and* whose sum of weights is the smallest.
- Examples:
  - – telephone switching network
  - – electronic board wiring
- The nodes and subset of connections form a tree!
- This tree is called the Minimum Spanning Tree (MST – עץ פורש מינימום)

Example: spanning tree



Cost: 51

Example: minimum spanning tree



Cost: 37

Spanning trees

- Definition: Let $G=(V,E)$ be a weighted connected undirected graph. A *spanning tree* of G is a subset $T \subseteq E$ of edges, such that the sub-graph $G'=(V,T)$ is connected and acyclic.
- The *minimum spanning tree* (MST) is a spanning tree that minimizes the sum:

Generic MST algorithm

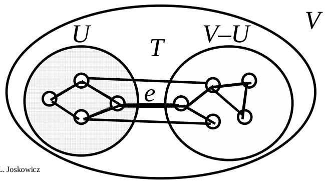
Greedy strategy: grow the minimum spanning tree one edge at a time, making sure that the added edge preserves the tree structure and the minimality condition→ add "safe" edges incrementally.

Generic-MST($G=(V,E)$)

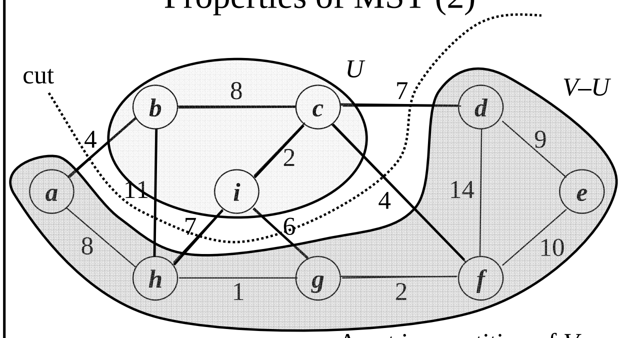
$T = \emptyset$;
while (T is not a spanning tree of G) **do**
  choose a *safe* edge $e=(u,v) \in E$
  $T = T \cup \{e\}$
return T

Properties of MST (1)

- Question: how to find *safe edges* efficiently?
- Theorem 1: Let and $e=(u,v)$ be a minimum weight edge with one endpoint in U and the other in $V–U$. Then there exists a minimum spanning tree T such that e is in T.



Properties of MST (2)



A cut is a partition of V

Properties of MST (2)

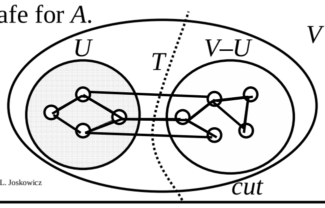
Proof: Let T be an MST. If e is not in T, add e to T. Because T is a tree, the addition of e creates a cycle which contains e and at least one more edge $e'=(u',v')$, where $u'\in U$ and $v'\in V–U$.

Clearly, $w(e) \leq w(e')$ since e is of minimum weight among the edges connecting U and $V–U$. We can thus delete e' from T.

The resulting $T' = T – \{e'\} \cup \{e\}$ is a tree whose weight is less or equal than that of T: $w(T') \leq w(T)$.

Properties of MST (3)

Theorem 2: Let $G=(V,E)$ be a connected undirected graph and A a subset of E included in a minimum spanning tree T for G. Let $(U, V–U)$ be a cut that respects A (no edge of A crosses the cut), and let $e=(u,v)$ be a minimum weight edge crossing $(U, V–U)$. Then e is safe for A.



Properties of MST (4)

Proof: Define an edge e to be a *light edge* crossing a cut if its weight is the minimum crossing the cut.

Let T be an MST that includes A, and assume T does not contain the light edge $e = (u,v)$ (if it does, e is safe).

Construct another MST T' that includes $A \cup \{e\}$. The edge forms a cycle with edges on the path p from u to v in T. Since u and v are on opposite sides of the cut, there is at least one edge $e' = (x,y)$ in T on the path p that also crosses the cut. The edge e' is not in A because the cut respects A. Since e' is on the unique path from u to v in T, removing it breaks T into two components.

Properties of MST (5)

Adding $e = (u,v)$ reconnects the two components to form a new spanning tree:

$$T' = T – \{e'\} \cup \{e\}$$

We now show that T' is a MST. Since $e = (u,v)$ is a light edge crossing $(U, V–U)$ and $e' = (x,y)$ also crosses this cut, $w(u,v) \leq w(x,y)$. Thus:

$$w(T') = w(T) – w(u,v) + w(x,y)$$
$$\leq w(T)$$

Since T is an MST and $w(T') \leq w(T)$, then $w(T') = w(T)$ and T' is also an MST.

Properties of MST (6)

Corollary: Let $G=(V,E)$ be a connected undirected graph and A a subset of E included in a minimum spanning tree T for G, and let $C = (V_C, E_C)$ be a tree in the forest $G_A = (V,A)$. If e is a light edge connecting C to some other component in G_A, then e is safe for A.

Proof: The cut $(V_C, V–V_C)$ respects A, and e is a light edge for this cut. Therefore, e is safe.

Two algorithms to find an MST

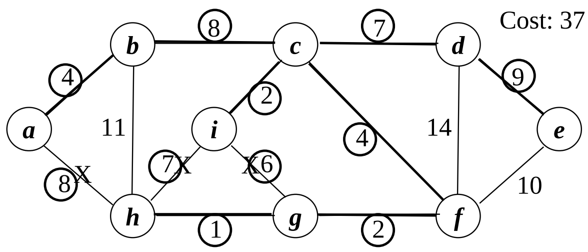
There are two ways of adding a safe edge:

1. Kruskal's algorithm: the set A is a forest and the safe edge added is always the least-weight edge in the graph connecting two distinct components (Theorem 2).
2. Prim's algorithm: the set A is a tree and the safe edge added is always the least-weight edge connecting A to a vertex not in A (Theorem 1).

Kruskal's algorithm

MST-Kruskal(G)
  $A \leftarrow \emptyset$
  for each vertex $v \in V$ **do**
    Make-Set(v)
  sort the edges in E in non-decreasing weight order
  for each edge $e = (u,v) \in E$ **do**
    if Find-Set(u) $\neq$ Find-Set(v) /* the trees are distinct */
    then
      $A \leftarrow A \cup \{e\}$
      Union(u,v) /* combine two trees */
  return A

Example: Kruskal's algorithm



Analysis of Kruskal's algorithm

Correctness: follows directly from Theorem 2.

Complexity: Depends on the implementation of the set operations! A naïve implementation takes $O(|V|\,|E|)$.

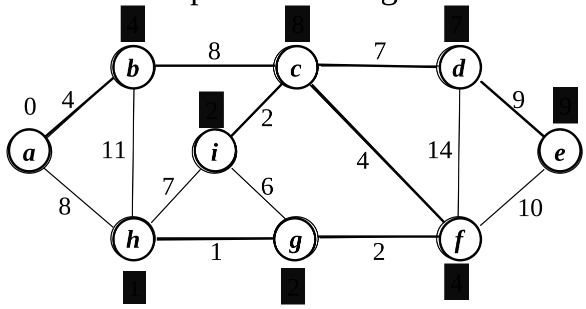
- Sorting the edges takes $O(|E| \lg |E|)$.
- the **for** loop goes over every edge and performs two Find-Set and one Union operation. These can be implemented to take $O(1)$ amortized time.

The total running time is $O(|E| \lg |E|) = O(|E| \lg |V|)$.

Prim's algorithm

MST-Prim(G, $root$)
  for each vertex $v \in V$ **do**
    $key(v) \leftarrow \infty$; $\pi[v] \leftarrow$ ***null***
  $key(root) \leftarrow 0$; $Q \leftarrow V$
  while Q is not empty **do**
   $u \leftarrow$ Extract-Min(Q)
   for each v that is a neighbor of u **do**
    if $v \in Q$ and $w(u,v) < key(v)$
     then $\pi[v] \leftarrow u$
       $key(v) \leftarrow w(u,v)$ /*decrease value of key */

Example: Prim's algorithm



Cost: 37

Analysis of Prim's algorithm

Correctness: follows directly from the Theorem 1.

Complexity: Depends on the implementation of the minimum priority queue. With a binary mean-heap, we have:

- Building the initial heap takes $O(|V|)$.
- Extract-Min takes $O(\lg |V|)$ per vertex → total $O(|V| \lg |V|)$
- The **for** loop is executed $O(|E|)$.
- Membership test is $O(1)$. Decreasing a key is $O(\lg |V|)$.

Overall, the running time is $O(|V| \lg |V| + |E| \lg |V|) = O(|E| \lg |V|)$.

Summary: MST

- MST is a tree of all nodes with minimum total cost
- Two greedy algorithms for finding MST:
  - Kruskal's algorithm: edge-based. Runs in $O(|V|\,|E|)$.
  - Prim's algorithm: vertex-based. Runs in $O(|E| \lg |V|)$.
- Complexity of Kruskal's algorithm can be improved with Union-Find ADT to $O(|E| \lg |V|)$,
- Complexity of Prim's algorithm can be improved with Fibonacci heaps to $O(|V| \lg |V| + |E|)$.
- Randomized algorithm has $O(|V| + |E|)$ expected time.