Minimum Spanning Trees

Data Structures and Algorithms (094224)

Yuval Emek

Winter 2022/23

1. Spanning trees in weighted graphs

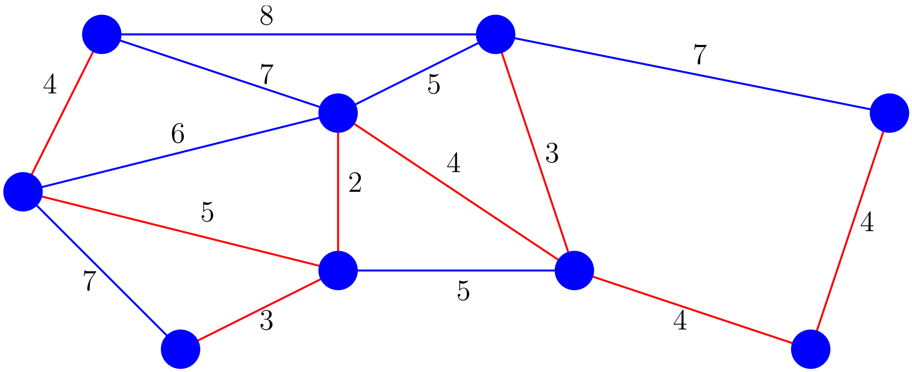
2. Prim's algorithm
   - Correctness
   - Run-time

3. Kruskal's algorithm
   - Correctness

Cheapest connected subgraph

- Connected undirected graph $G = (V, E)$ with edge weight function $w : E \to \mathbb{R}$
- $H = (V, T)$ is a *spanning tree* ⟨עץ פורש⟩ of G if
  - H is a (spanning) subgraph of G, i.e., $T \subseteq E$; and
  - H is a tree
- Often address the spanning tree by its edge set T
  - Formally wrong!
  - Little risk for ambiguity because the vertex set is known
- $T \subseteq E$ is a *minimum spanning tree (MST)* ⟨עץ פורש מינימום⟩ if
  - T is a spanning tree of G; and
  - $w(T) \leq w(T')$ for all spanning trees T' of G
    - Recall that $w(F) = \sum_{e \in F} w(e)$
- The minimum spanning tree problem: construct an MST for a given weighted undirected graph
- Motivation: the "cheapest" subset of edges that ensures connectivity

MST — example



Growing a good subset

- Edge subset $F \subseteq E$ is *good* if there exists an MST $T \supseteq F$
- Edge $e \in E - F$ is *safe* for (good) F if $F \cup \{e\}$ is still good
- Main idea: grow a good subset by iteratively adding safe edges

  1: $F = \emptyset$
  2: **while** F is not an MST **do**
  3: find a safe edge e for F
  4: $F = F \cup \{e\}$

  - How many iterations? n-1 (because T is a tree)
- The challenge: find a safe edge

Cuts

- Consider some vertex subset $\emptyset \subset S \subset V$
- The (bi-)partition of V into $\{S, V - S\}$ is referred to as a *cut* ⟨חתך⟩
- Let $E(S, V - S)$ be the set of edges $(u, v) \in E$ such that

$$(u \in S \wedge v \in V - S) \quad \vee \quad (v \in S \wedge u \in V - S)$$

  - I.e., $|\{u, v\} \cap S| = 1$
  - The edges in $E(S, V - S)$ are said to *cross* ⟨חוצות⟩ cut $\{S, V - S\}$
- Edge $e \in E$ is *light* ⟨קלה⟩ for cut $\{S, V - S\}$ if
  - $e \in E(S, V - S)$; and
  - $w(e) \leq w(e')$ for every edge $e' \in E(S, V - S)$
- Cut $\{S, V - S\}$ *respects* edge subset $F \subseteq E$ if $F \cap E(S, V - S) = \emptyset$
  - I.e., no edge in F crosses cut $\{S, V - S\}$

Light edges are safe

Theorem

Let $F \subseteq E$ be a good edge subset and let $\{S, V - S\}$ be a cut that respects F. If $e \in E$ is light for $\{S, V - S\}$, then e is safe for F.

- Let $e = (u, v)$, where $u \in S$ and $v \in V - S$
- $\{S, V - S\}$ respects F, hence $e \notin F$
- Let $T \subseteq E$ be an MST such that $F \subset T$
  - Exists because F is good and (V, F) is not connected (why?)
- If $e \in T$, then we are done, so assume that $e \notin T$
- Let P be the unique simple (u, v)-path in (V, T)
- Since P leads from $u \in S$ to $v \in V - S$, it must contain some edge $e' = (u', v')$ such that $u' \in S$ and $v' \in V - S$
- $w(e') \geq w(e)$ as e is light for $\{S, V - S\}$

Light edges are safe — cont.

- Let $T' = T \cup \{e\} - \{e'\}$
- Claim 1: (V, T') is connected
  - Consider some $x, y \in V$
  - Let Q be the unique simple (x, y)-path in (V, T)
    - T is an MST and in particular a tree
  - If $e' \notin Q$, then Q is also an (x, y)-path in (V, T')
  - Otherwise, augment Q with $(P - \{e'\}) \cup e$ which is a (u', v')-path in (V, T')
- $|T'| = |T| = |V| - 1$
- $\Longrightarrow (V, T')$ is a tree
- Claim 2: T' is an MST
  - T is an MST
  - $w(T') = W(T) + w(e) - w(e') \leq w(T)$
- Since $F \cup \{e\} \subseteq T'$, it follows that e is safe for F $\blacksquare$

The Prim algorithm

- Input:
  - Connected undirected graph $G = (V, E)$
  - Weight function $w : E \to \mathbb{R}$
- Output: an MST T of G
- Representing T:
  - Root T at an arbitrary vertex $r \in V$
  - Each vertex $v \in V - \{r\}$ holds a pointer $v.\pi$ (predecessor) to its parent
  - Upon termination, $T = \{(v.\pi, v) \mid v \in V - \{r\}\}$

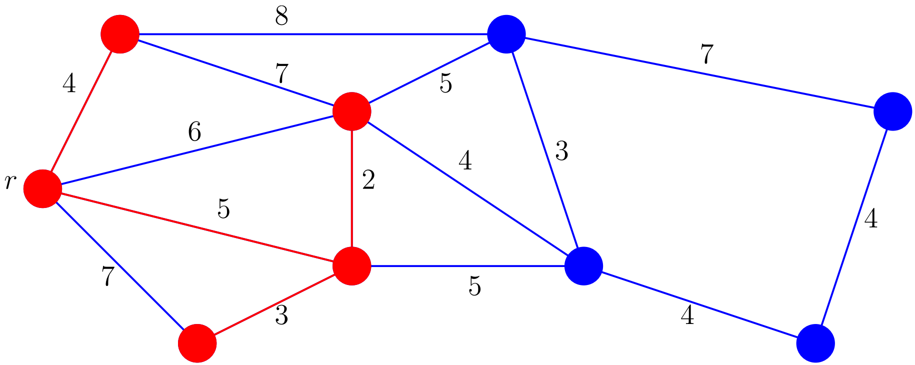
Pseudocode

$\texttt{Prim}(G, w)$

for all $u \in G.V$ **do**
    $u.\pi = NIL$
    $u.key = \infty$
pick an arbitrary vertex $r \in G.V$
$r.key = 0$
$Q = G.V$
while $Q \neq \emptyset$ **do**
    $u = \texttt{Extract\_Min}(Q)$ ▷ minimum w.r.t. *key*
    for all $v \in G.Adj[u]$ **do**
        if $v \in Q$ and $w(u, v) < v.key$ **then**
            $v.\pi = u$
            $v.key = w(u, v)$

- Resemblance to Dijkstra's algorithm

Prim — illustration



1. Spanning trees in weighted graphs
2. Prim's algorithm
   - Correctness
   - Run-time
3. Kruskal's algorithm
   - Correctness

Growing a tree

- The algorithm (implicitly) maintains:
  - Vertex subset $S = V - Q$
  - Edge subset $F = \{(v.\pi, v) \mid v \in S - \{r\}\}$

Lemma

(S, F) is a tree.

- (S, F) is a well defined graph as $u = v.\pi$ implies that $u \in S$
  - $v.\pi$ can be set to u (line 11) only after u is extracted from Q (line 8)
- (S, F) doesn't admit π-oriented cycles
  - $u = v.\pi$ implies that u joined S before v
- (S, F) is cycle free
  - An unoriented cycle implies that some vertex has > 1 predecessors
- $|F| = |S| - 1$ ∎

Corollary

The algorithm returns a spanning tree.

Picking light edges

Corollary

$\{S, V - S\}$ respects F.

Observation

still in Q

At the end of each iteration of the while loop (lines 7–12), if $v \in V - S$ and $N(v) \cap S \neq \emptyset$, then $v.key = \min\{w(u,v) \mid u \in N(v) \cap S\}$.[a]

[a] $N(v)$ denotes the set of v's neighbors

Corollary

When vertex $u \neq r$ is extracted from Q (line 8), edge $(u.\pi, u)$ is light for $\{S, V - S\}$.

Corollary

The algorithm returns an MST.

1. Spanning trees in weighted graphs
2. Prim's algorithm
   - Correctness
   - Run-time
3. Kruskal's algorithm
   - Correctness

Run-time analysis

- The initialization takes $O(n)$ time
- Each vertex is extracted from Q (line 8) exactly once
- $\Longrightarrow$ Each edge is examined (lines 10–12) exactly twice
- A naive implementation:
  - Each call to `Extract_Min` takes $O(n)$ time
  - Run-time: $O(n^2 + m) = O(n^2)$
- Implementing Q using a binary heap:
  - Each call to `Extract_Min` takes $O(\log n)$ time
  - Each call to `Decrease_Key` (hidden in line 12) takes $O(\log n)$ time
  - Run-time: $O((n + m) \log n)$
- Implementing Q using a Fibonacci heap:
  - Run-time: $O(n \log n + m)$
  - Beyond the scope of this course

1. Spanning trees in weighted graphs

2. Prim's algorithm
   - Correctness
   - Run-time

3. Kruskal's algorithm
   - Correctness

The Kruskal algorithm

- Input:
  - Connected undirected graph $G = (V, E)$
  - Weight function $w : E \to \mathbb{R}$
- Output: an MST T of G

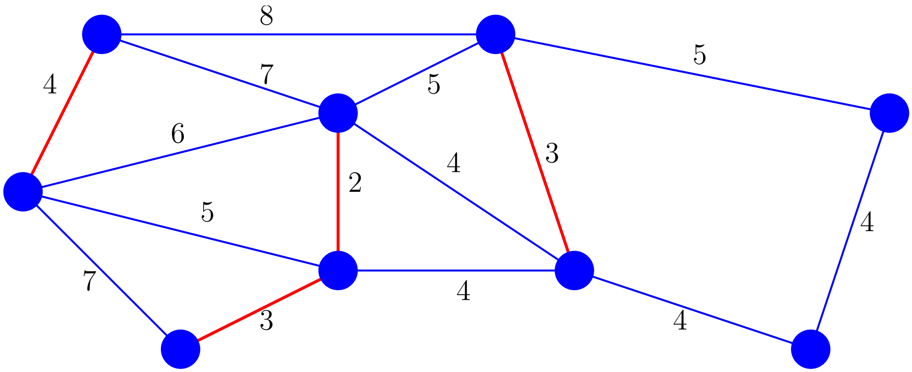
Pseudocode

Kruskal(G, w)

```
1: copy the edges in G.E into array A[1...m]
2: sort A w.r.t. the edge weights
3: T = ∅
4: for i = 1,...m do
5:     e = A[i]
6:     if (G.V, T ∪ {e}) is cycle free then
7:         T = T ∪ {e}
8: return T
```

- Missing details: implementation of the if condition in line 6
    - Less important since we don't focus on the algorithm's run-time

Kruskal — illustration



Growing a forest

Observation

(V, T) is a forest (cycle free) at all times.

Corollary

The algorithm returns a spanning tree.

- Assume by contradiction: T is not connected upon termination
- $\Longrightarrow$ T is respected by some cut $\{S, V - S\}$
- $\{S, V - S\}$ is crossed by some edge $e \in E$
  - G is connected
- e should have been added to T during its respective iteration $_{(\rightarrow\leftarrow)}$ $\blacksquare$

The trees of the forest

- Consider the beginning of iteration i of the for loop (lines 4–7)
  - Let $e = A[i]$ (line 5)
  - Let $C_1, \ldots C_k$ be the connected components (trees) of the forest (V, T)

Observation

The if condition (line 6) is satisfied iff $e \in E(C_i, V - C_i)$ for some $1 \leq i \leq k$.

Observation

If $e \in E(C_i, V - C_i)$ for some $1 \leq i \leq k$, then e is light for $\{C_i, V - C_i\}$.

Corollary

If e is added to T, then e is safe for T.

Corollary

The algorithm returns an MST.