Algorithm Theory

18 Minimum Spanning Trees

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08
Minimum Spanning Trees

Undirected graph \( G = (V, E) \)

Weight function \( c : E \rightarrow R \)

Let \( T \subseteq E \) be a tree (connected acyclic subgraph)

Total weight of \( T \) :
\[
c(T) = \sum_{(u,v) \in T} c(u, v)
\]
Minimum Spanning Trees

A tree $T \subseteq E$ connecting all nodes in $V$ with minimum weight is called a minimum spanning tree.
Growing a MST

- **Invariant:**
  - Maintain a set $A \subseteq E$ that is a subset of some minimum spanning tree

- **Definition:**
  - An edge $(u,v) \in E \setminus A$ is a safe edge for $A$ if $A \cup \{(u,v)\}$ is also a subset of some minimum spanning tree
A cut \((S, V \setminus S)\) is a partition of \(V\).

An edge \((u,v)\) crosses \((S, V \setminus S)\), if one of its endpoints is in \(S\) and the other is in \(V \setminus S\).
Greedy Approach

Algorithm Generic-MST(G,w)
1. A ← ∅
2. while A does not form a spanning tree do
3. Find an edge (u,v) that is safe for A;
4. A ← A ∪ {(u,v)};
5. endwhile;
A cut respects a set $A$ of edges if no edges in $A$ crosses the cut.
Cuts

An edge is a **light edge crossing a certain cut** if its weight is the minimum of any edge crossing the cut.
Safe Edges

**Theorem:**

Let $A$ be a subset of some minimum spanning tree $T$, and let $(S, V \setminus S)$ be a cut that respects $A$. If $(u,v)$ is a light edge crossing $(S, V \setminus S)$ then $(u,v)$ is safe for $A$.

**Proof**

Case 1: $(u,v) \in T$: o.k.

Case 2: $(u,v) \notin T$:

We construct another minimum spanning tree $T'$ with $(u,v) \in T$ and $A \subseteq T'$. 
Safe Edges

- Adding \((u,v)\) to \(T\) yields a cycle.
  - On this cycle, there is at least one edge \((x,y)\) in \(T\) that also crosses the cut

\[ T' = T \setminus \{(x,v)\} \cup \{(u,v)\} \text{ is a minimum spanning tree, since} \]

\[ w(T') = w(T) - w(x,y) + w(u,v) \leq w(T) \]
The Graph $G_A$

- $G_A = (V, A)$
  - is a forest, i.e. a collection of trees
  - at the beginning, when $A = \emptyset$, each tree consists of a single vertex
  - any safe edge for $A$ connects distinct trees

Corollary
- Let $B$ be a tree in $G_A = (V, A)$, If $(u, v)$ is a light edge connecting $B$ to some other tree in $G_A$, then $(u, v)$ is safe for $A$.

Proof:
- $(B, V \setminus B)$ respects $A$ and $(u, v)$ is a light edge for this cut
Kruskal’s Algorithm

Always choose an edge of smallest weight that connects two trees $B_1$ and $B_2$ in $G_A$
Algorithm of Kruskal

1. \( A \leftarrow \emptyset; \)
2. for all \( v \in V \) do \( B_v \leftarrow \{ v \}; \) endfor;
3. Generate a list \( L \) of all edges in \( E \), sorted in non-decreasing order of weight
4. for all \( (u,v) \) in \( L \) do
5. \( B_1 \leftarrow \text{FIND}(u); \) \( B_2 \leftarrow \text{FIND}(v); \)
6. if \( B_1 \neq B_2 \) then
7. \( A \leftarrow A \cup \{(u,v)\}; \) \( \text{UNION} (B_1, B_2, B_1); \)
8. endif;
9. endfor;

Running time: \( O( m \alpha(m,n) + m + m \log m ) = O( m \log m ) \)
Prim’s Algorithm

A is always a single tree.
Start from an arbitrary root vertex \( r \).
In each step, add a light edge to \( A \) that connects \( A \) to a vertex in \( V \setminus A \).
Implementation

Q: Priority Queue containing all vertices $v \in V \setminus A$.

key of vertex $v$: minimum weight of any edge connecting $v$ to a vertex in $A$

For a node $v$, let $p[v]$ denote the parent of $v$ in the tree.

$A = \{(v, p[v]): v \in V \setminus (\{r\} \cup Q)\}$

$r = \text{root vertex}$
Prim's Algorithm

1. for all \( v \in V \) do Insert\((Q, \infty, v)\); endfor;
2. Choose a root vertex \( w \in V \);
3. DecreaseKey\((Q, 0, w)\); \( p[w] \leftarrow \text{nil} \);
4. while \( \neg \text{Empty}(Q) \) do
5. \((d, u) \leftarrow \text{DeleteMin}(Q)\);
6. for all \((u,v) \in E\) do
7. if \( v \in Q \) and \( w(u,v) < \text{key of } v \) then
8. DecreaseKey\((Q, w(u,v), v)\); \( p[v] \leftarrow u \);
9. endif;
10. endfor;
11. endwhile;

Running time: \( O(n \log n + m) \)
Algorithm Theory
18 Minimum Spanning Trees

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08