Minimum Spanning Trees

Jonathan Backer
backer@cs.ubc.ca

Department of Computer Science
University of British Columbia

June 24, 2007

Introduction

Reading:

- CLRS: “Graph Representations”, 22.1
  CLRS: “Minimum Spanning Trees”, 23
- GT: “Graphs”, 6.1-6.2
  GT: “Minimum Spanning Trees”, 7.3

We will minimize the cost of connecting a set of objects together. Typical examples include wiring a network and building roads.

Prim’s algorithm is a prime example of a greedy algorithm and we will use the same style of argument to prove it’s correctness.

Graphs

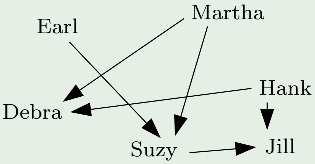
Definition

A graph $G(V, E)$ is a set of vertices V that are joined by edges $E \subseteq V \times V$. The vertices represent objects and edges represent a binary relation.

We assume that V is finite, every edge from a vertex leads to a different vertex (no loops), and there is at most one edge from one vertex to another vertex (no multiple edges).

Example: Heredity

Vertices are people and an edge from u to v denotes that u is a child of v.

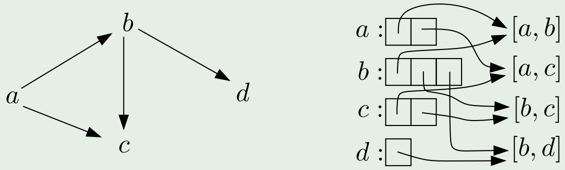


What properties do heredity graphs have?

Adjacency lists

- Each edge has pointers to its endpoints.
- Each vertex has a list of pointers to incident edges.

Example

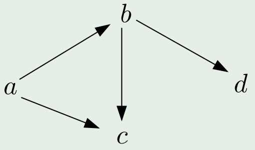


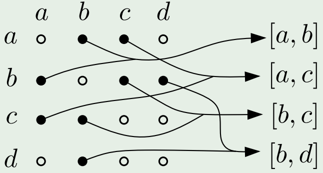
- Easy to find neighbours of a vertex.
- Takes $O(|V| + |E|)$ space.
- Testing if two vertices are joined takes $O(|V|)$ time.

Adjacency matrices

- Each edge has pointers to endpoints.
- A $|V| \times |V|$ matrix A refers to edges:
  - $A[u][v]$ points to the edge from u to v.

Example



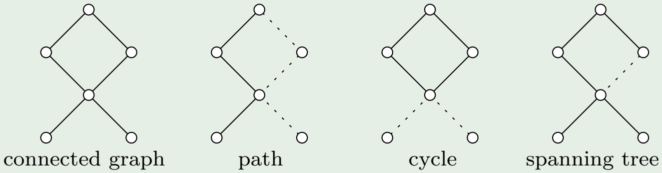


- Testing if u and v are joined takes $\Theta(1)$.
- Finding all neighbours takes $O(|V|)$ time.
- Requires $O(|V|^2)$ space.

Connectivity

- A path from u to v is a sequence of vertices $u = x_0, x_1, \ldots, x_t = v$ where consecutive vertices are adjacent.
- A cycle is a path that starts and ends at the same vertex.
- An undirected graph is connected if there is path between every pair of distinct vertices.
- A tree is a connected, undirected graph with no cycles.
- A spanning tree of a graph $G(V, E)$ is a tree $T(V, E')$ where $E' \subseteq E$.

Example



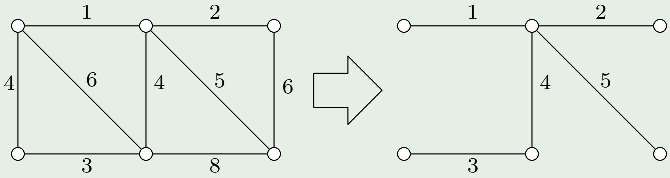
Minimum spanning trees

Problem

We are given a connected, undirected graph $G(V, E)$ with edge weights $w : E \rightarrow \mathbb{R}^{\geq 0}$.

Find a spanning tree $T(V, E')$ with the smallest total weight $\sum_{e \in E'} w(e)$.

Example



Prim's algorithm (sketch)

- Start from a fixed vertex (v_1)
- Iteratively add the vertex that is cheapest to reach from the vertices that we have spanned so far.

$$
\begin{array}{l}
\texttt{Algorithm Prim}(V,\ E,\ w) \\
\quad T \leftarrow \emptyset \\
\quad S \leftarrow \{v_1\} \\
\quad \texttt{while } S \neq V \texttt{ do} \\
\qquad \texttt{find } e = \{u, v\} \texttt{ of minimum weight such that} \\
\qquad\qquad u \in S \texttt{ and } v \in V \setminus S \\
\qquad T \leftarrow T \cup \{e\} \\
\qquad S \leftarrow S \cup \{v\} \\
\quad \texttt{return } T
\end{array}
$$

- An $O(|V| \times |E|)$ runtime complexity as written.
- Use a priority queue (heap) to make the `find` step fast!

Initialization

- *cost* is the current cheapest cost of adding the vertex to the MST using an edge with one endpoint already spanned
- *prev* is the other endpoint of the edge that gives us the lowest cost

```
Algorithm Prim(V, E, w)
    cost[v_1] ← 0
    prev[v_1] ← ∅
    spanned[v_1] ← false
    Q.add(v_1,0)

    for i ← 2 to n do
        cost[v_i] ← ∞
        prev[v_i] ← ∅
        spanned[v_i] ← false
        Q.add(v_i,∞)
```

Main loop

```
// greedy loop
for i ← 1 to |V| do
    v ← Q.deleteMin()
    spanned[v] ← true
    if prev[v] ≠ ∅ then
        add {v, prev[v]} to the MST
    for each neighbour n of v do
        if spanned[n] = false and
            w({n, v}) < cost[n] then
                cost[n] ← w({n, v})
                prev[n] ← v
                Q.updatePriority(n, cost[n])
```

Runtime complexity

Focus on priority queue operations:

- each vertex added to the queue once:
  $|V| \times O(\log |V|) = O(|V| \log |V|)$
- each vertex removed from the queue once:
  $|V| \times O(\log |V|) = O(|V| \log |V|)$
- priority update at most once for every edge:
  $|E| \times O(\log |V|) = O(|E| \log |V|)$

Total cost: $O([|V| + |E|] \log |V|)$

Correctness

Theorem

Prim's algorithm correctly computes a minimum spanning tree.

Proof

Let $T(k)$ be the tree constructed after adding k edges. Our proof is inductive on k. We maintain the property that $T(k)$ can be extended into a MST. The tree $T(0)$ can trivially be extended into an MST.

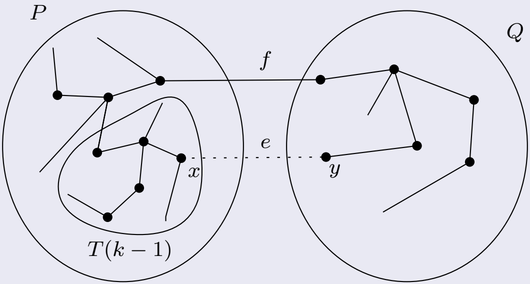
Assume that $T(k-1)$ is a subtree of some MST R. We use R to construct a MST S such that $T(k)$ is a subtree of S. Let e be the edge added to $T(k-1)$ to get $T(k)$. If $e \in R$, then R is the desired S. So suppose not.

Now $U \cup \{e\}$ has a cycle because $e \notin R$. Some edge f of this cycle (other than e) leaves $T(k-1)$.

Prim's correctness (cont'd)

Proof

Removing f breaks R into two trees P and Q. One of these trees (say P) contains $T(k-1)$.



Let $S = (R \setminus \{f\}) \cup \{e\}$. It has $|V| - 1$ edges. We now argue that it is connected to prove that it is a tree. Consider $u, v \in V$.

Case 1: If the u, v-path in R doesn't use f, then it is a path in U.

Prim's correctness (cont'd)

Proof

Case 2: Otherwise assume without loss of generality that $u \in P$ and $v \in Q$. Let $e = \{x, y\}$ where $x \in T(k-1)$. Then there is a u, x-path in P and a y, v-path in Q that we can bridge with e to get a u, v-path in S.

So S is a tree. Moreover $w(e) \leq w(f)$ because e has the smallest weight of all edges leaving $T(k-1)$. So the total weight of S is no greater than the weight of R. Hence S is a MST containing $T(k)$. $\square$