Algorithm Design Techniques

- Induction (or Self-Reduction)
- Reduction
- Divide and Conquer (special case of Self-Reduction)
- Greedy

Greedy Paradigm
Get what you can NOW!

But sometimes it’s better to look around!

But sometimes it isn’t …

I hate gas stations!

- I’m driving cross country and my route is fixed.
- My map tells me exactly where every gas station along the route is located.
- I want to minimize the number of times I stop for gas…
- … without running out!

Greedy

- First stop
  – I’ll stop at the farthest gas station I can get to without running out.
- Then repeat
Greedy is Optimal!
• Can the optimal make a first stop that is later?

Minimum Spanning Tree
• Input: Weighted graph G
• Output: Minimum weight spanning tree of G

Weighted Graph
• \( G=(V,E) \) is a connected, weighted graph with \( n \) vertices and \( m \) edges.

Spanning Tree
• A spanning tree of \( G \) is a connected, acyclic subgraph with vertex set \( V \).

Weight of Spanning Tree
• The weight of spanning tree of \( G \) is the sum of the weights of its edges.

Minimum Spanning Tree
• A minimum spanning tree of \( G \) is one with smallest possible weight.
• Find an MST of the following graph:
Prim’s Algorithm

Choose a vertex \( w \in V \)

\( F = \{ w \} \)

While \( V - F \neq \emptyset \)

Let \( e \) be a minimum weight edge that emerges from \( F \)

\( F = F + \{ e \} \)

Prim’s example

Start with red vertex

Prim’s example
Prim’s example

Prim’s example

Prim’s example

Prim’s Algorithm

• Is it correct?
• Is it efficient?

Cut

• A cut is a partition of the vertices of G into two sets (R, B).
• An edge e crosses the cut if it has an endpoint in each set of the cut.
• Which edges cross the (R, B) cut?

Cut Theorem

• Let (R, B) be a cut of graph G and let e be a minimum weight edge crossing the cut.
• Then e is in some minimum spanning tree of T.
• If e is the least weight edge spanning the cut then it is in every minimum spanning tree of T.
Tree Facts

• A tree on n nodes has n-1 edges.

• If e is an edge of T then T-{e} is a forest consisting of two trees.

• If e is an edge of G but not of T then T+{e} contains exactly one cycle.

Proof: Cut Theorem

• Let (R,B) be a cut of graph G and let e be a minimum weight edge crossing the cut.
• Let T be a minimum spanning tree of G.
• If e is in T we are done so suppose not.
• Consider the graph T+{e}.

Proof: Cut Theorem

• Consider the graph T+{e}.
  – By our tree facts this graph has exactly one cycle and the cycle includes e.
  – Removing any edge of the cycle yields a spanning tree of G.
  – If the cycle contains an edge e’ such that w(e’) ≥ w(e) then T+{e}-{e'} is a spanning tree with weight no more than T.
Consider the cut \((R,B)\)

- Since \(e\) spans the cut at least one other edge of the cycle must also span the cut.
- Why?
- So what?

Prim’s Algorithm

- Is it correct?
  - If the edge weights are unique then it follows immediately from the cut theorem.
  - What if the edge weights are not unique?
- Is it efficient?

Prim’s Algorithm

Choose a vertex \(w \in V\)
\(F=\{w\}\)
While \(V-F \neq \phi\)
  Let \(e\) be a minimum weight edge that emerges from \(F\)
  \(F=F+\{e\}\)

Running Time: \(O(n?)\)

Choose a vertex \(w \in V\)
\(F=\{w\}\)
While \(V-F \neq \phi\)
  Let \(e\) be a minimum weight edge that emerges from \(F\)
  \(F=F+\{e\}\)

How should we implement this?

Prim’s Algorithm

Running Time: \(O(nm)\)

Choose a vertex \(w \in V\)
\(F=\{w\}\)
While \(V-F \neq \phi\)
  Let \(e\) be a minimum weight edge that emerges from \(F\)
  \(F=F+\{e\}\)

Naïve approach \(O(m)\)
Fringe vertex

A fringe vertex $v$ is a fringe vertex if it is in $V - F$ and it is connected by an edge to a vertex $u$ in $F$.

A less naïve approach …

List of fringe vertices and for each its minimum weight edge to $F$:

- $[b,2]$,
- $[d,3]$,
- $[a,5]$,
- $[c,10]$.

And even better…

• Keep the fringe vertices in a heap.

Prim’s Algorithm

A Better Implementation

Choose a vertex $x \in V$

$F = \{x\}$, $H = \phi$

For each $e = (u,x)$: Add record $[u,e]$ to heap $H$ keyed on $w(e)$

While $H \neq \phi$

- $[u,e] = \text{Find-min}(H)$
- Add $u$ and $e$ to $F$
- For each edge incident to $u$: Update heap

Update heap:

- $[b,2]$: remove $[b,2]$ from heap
- Add new fringe vertices: $[d,3],[a,5],[c,10],[e,12]$
- Update edge weights: $[d,3],[a,5],[c,9],[e,12]$
Are these standard operations?

- Extract-min
- Add element to heap
- Reduce key of element in heap ??!

Decrease key

- Next homework assignment: Design decrease key algorithm for heaps that runs in time $O(\log(n))$.

Prim’s Algorithm

Running Time

Choose a vertex $x \in V$
$F = \{x\}, H = \emptyset$

For each $e = (u, x)$: Insert

While $H \neq \emptyset$

- $[u, e]$: Extract-min($H$)
- Add $u$ and $e$ to $F$

For each edge incident to $u$: Insert or Decrease-key or do nothing

but wait … suppose we could decrease-key in time $O(1)$

- Heap operations across algorithm:
  - $n$ Extract-mins $O(\log(n))$ each
  - $n$ Inserts $O(\log(n))$ each
  - $m-n$ Decrease-keys $O(\log(n))$ each
  - $m$ Do nothings $O(1)$ each
- Then total time is $O(m \log(n))$

Bravo, bravo …
**Kruskal’s (Greedy) Algorithm**

Let \( e_1, e_2, \ldots, e_m \) be the edges of \( G \) sorted by increasing weight.

- **F** = \( V \) (\( F \) is a forest of isolated vertices)
- For \( i = 1 \) to \( m \)
  - If \( F + \{e_i\} \) is acyclic then \( F = F + \{e_i\} \).
- Return(\( F \))

**Kruskal’s Algorithm**

- Order the edge weights. (In this graph the weights are unique.)
- \( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 \)

**Kruskal’s Algorithm-cont.**

- \( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 \)

**Kruskal’s Algorithm-cont.**

- \( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 \)
Kruskal’s Algorithm-cont.

• $1,2,3,4,5,6,7,8,9,10,11,12,13$

Kruskal’s Algorithm-cont.

• $1,2,3,4,5,6,7,8,9,10,11,12,13$

Can we add the edge with weight 7?

Kruskal’s Algorithm-cont.

• $1,2,3,4,5,6,7,8,9,10,11,12,13$

Can we add the edge with weight 8?

Kruskal’s Algorithm-cont.

• $1,2,3,4,5,6,7,8,9,10,11,12,13$

MST of $G$ with cost ______
Kruskal’s Algorithm

• Does it work in general?

• Prove it.

Kruskal’s Algorithm

Claim:
At each stage of the algorithm F is a subgraph of some MST of G.

Proof of Correctness

Claim: F is a subgraph of some MST of G.

Proof:
Consider the kth execution of the loop. Let T be a MST of G containing F. What can happen during the loop?
1. \( e_k \) is not added to F
2. \( e_k \) is added to F

Loop Invariant

F is a subgraph of some MST of G.

Proof:

1. \( e_k \) is not added to F
   In this case F does not change so the claim holds when execution of loop concludes.
Kruskal’s Algorithm
Proof of Correctness

Loop Invariant:
F is a subgraph of some MST of G.

Proof
1. e is added to F
   We know that T is a MST of G and T contains F.
   Need to show there is a MST of G that contains F+{e}
   If e is an edge of T we are done. So assume not.

What do we know?
- Assume e = (u,v). The vertices u and v are in separate connected components. Let S be the vertices of F.
- e is a minimum weight edge spanning (S, V-S)

Using our tree facts
- The graph T+{e} contains exactly one cycle.
- This cycle contains e and at least one additional edge e that spans (S, V-S).
- T+{e} - {e} is an MST of G.

Moreover
- T+{e} - {e} is an MST of G that contains the edges of F+{e}.

Running Time
- We’ll save that for later…