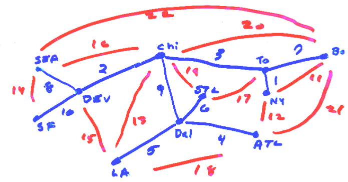
Minimum Cost Spanning Trees

Kruskal's Algorithm:

Another Example of the Greedy Method

Min Cost Spanning Tree

- Given: Undirected graph G = (V, E) & positive edge cost/weight $c(e) \in R$ for each $e \in E$.
- Find: connected $T \subseteq E$ minimizing $\Sigma_{e \in T}\, c(e)$



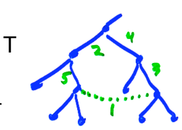
Applications

- Broadcast tree in a network
- Building roads or power lines
- Routing power & ground on a PC board
- Clustering
- …

Lemma 1: Trees and Cycles

Adding an edge to a tree creates a cycle; deleting any cycle edge gives a tree

- Corollary 1: Solution to MST is a tree
- Corollary 2: Cheapest edge in E is in some MST T
- Excercises:
  - 2^{nd}-cheapest edge also in T
  - 3^{rd}-cheapest?



Kruskal's Algorithm

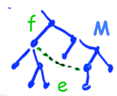
sort edges
K := ∅
while |K| < n-1 do
  e := next cheapest edge
  if K ∪ {e} is acyclic
    then K := K ∪ {e}
    else discard e

Correctness

Thm: Kruskal's algorithm builds an MST
Proof:

- Suppose Kruskal picks the tree K
- Suppose MST M maximizes |K ∩ M| among all MST
- For sake of contradiction, suppose K ≠ M
- Let e be the cheapest edge in K – M
- Then…

Claim



M ∪ {e} has a cycle containing an edge f s.t.

(1) f ∉ K, and

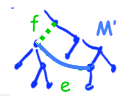
(2) c(f) ≥ c(e)

Proof:

(1) If all the cycle edges were in K, then K wouldn't be a tree.

(2) If c(f) < c(e), greedy looked at f before e. But {e' ∈ K | c(e') < c(e)} ∪ {f} ⊆ M, hence acyclic, so f would have been picked, but it wasn't.

Correctness (cont.)



Theorem: Kruskal's algorithm builds an MST

Proof:

- Suppose Kruskal picks the tree K
- Suppose MST M maximizes |K ∩ M| among all MSTs
- For sake of contradiction, suppose K≠M
- Let e be the cheapest edge in K – M
- From claim, ∃ f in cycle s.t. f ∉ K, c(f) ≥ c(e)
- Let M' = (M ∪ {e}) – {f}
- Then M' is an MST with |K ∩ M'| > |K ∩ M|. Contradiction. QED

Implementation

Testing "if K∪ {e} is acyclic":

Union/Find problem

Linear space

Time n α(n)

α(n) < 5 for all n < age of universe

Overall Timing

- O(e log e), where e = # edges
- The initial edge sort dominates
- Asymptotically faster algorithms are known, but tend to be complex enough that they may not be better in practice