CS38
Introduction to Algorithms

Lecture 3

April 8, 2014

Outline

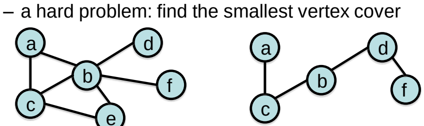
- greedy algorithms…
  - Dijkstra's algorithm for single-source shortest paths
- guest lecturer (this lecture and next)
  - coin changing
  - interval scheduling
  - MCST (Prim and Kruskal)

Greedy algorithms

- Greedy algorithm paradigm
  - build up a solution incrementally
  - at each step, make the "greedy" choice

Example: in undirected graph G = (V,E), a vertex cover is a subset of V that touches every edge

- a hard problem: find the smallest vertex cover



Dijkstra's algorithm

- given
  - directed graph G = (V,E) with non-negative edge weights
  - starting vertex s ∈ V
- find shortest paths from s to all nodes v
  - note: unweighted case solved by BFS

Dijkstra's algorithm

- shortest paths exhibit "optimal substructure" property
  - optimal solution contains within it optimal solutions to subproblems
  - a shortest path from x to y via z contains a shortest path from x to z
- shortest paths from s form a tree rooted at s
- Main idea:
  - maintain set S ⊆ V with correct distances
  - add nbr u with smallest "distance estimate"

Dijkstra's algorithm

```
Dijkstra(G = (V,E), s)
S = ∅, s.dist = 0, build Min-Heap H from V, keys are distances
while H is not empty
    u = EXTRACT-MIN(H)          ← "greedy choice"
    S = S ∪ {u}
    for each neighbor v of u
        if v.dist > u.dist + weight(u,v) then
            v.dist = u.dist + weigth(u,v), DECREASE-KEY(H, v)
```

Lemma: can be implemented to run in O(m) time plus n EXTRACT-MIN and m DECREASE-KEY calls.

Proof?

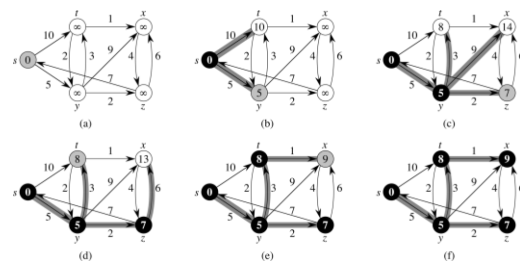
Dijkstra's algorithm

```
Dijkstra(G = (V,E), s)
S = ∅, s.dist = 0, build Min-Heap H from V, keys are distances
while H is not empty
    u = EXTRACT-MIN(H)          ← "greedy choice"
    S = S ∪ {u}
    for each neighbor v of u
        if v.dist > u.dist + weight(u,v) then
            v.dist = u.dist + weigth(u,v), DECREASE-KEY(H, v)
```

Lemma: can be implemented to run in O(m) time plus n EXTRACT-MIN and m DECREASE-KEY calls.

Proof: each vertex added to H once, adj. list scanned once, O(1) work apart from min-heap calls

Dijkstra's example from CLRS



Dijkstra's algorithm

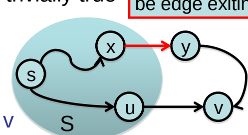
Lemma: invariant of algorithm: for all v ∈ S it v.dist = distance(s, v).

Proof: induction on size of S

– base case: S = ∅, trivially true

– case |S| = k:

consider any other s – v path, let (x,y) be edge exiting S



x.dist, u.dist correct by induction, so s – y path already longer than s – v since algorithm chose latter

Dijkstra's algorithm

- We proved:

Theorem (Dijkstra): there is an O(n + m log n) time algorithm that is given
  a directed graph with nonnegative weights
  a starting vertex s
and finds
  distances from s to every other vertex
  (and produces a shortest path tree from s)

– later: Fibonacci heaps: O(n log n + m) time