EECS 336: Introduction to Algorithms	**Lecture 6**
Dynamic Greedy	Dijkstra, Prim

Reading: 4.4.

Last Time:

- greedy-by-value
- MST
- matroid

Today:

- dynamic greedy
- shortest paths, MSTs

Dynamic Greedy Algorithms

"adjust ordering dynamically as greedy algorithm proceeds"

Template: Repeat:

- Process minimal element by metric.
- Adjust metric on remaining elements.

Note: priority queues useful for dynamic greedy algs.

Def: priority queue data structure

Operations:

- insert(v,k): adds elt v to queue with key k (priority)
- decreasekey(v,k): decreases the key of v to k

  (if key is less than k, leave it the same)
- deletemin: returns elt with minimum key.

Runtimes:

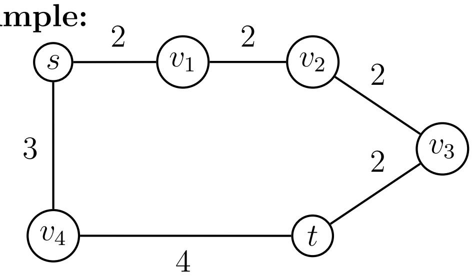
- can implement all operations in $O(\log n)$

Shortest Paths

"find short path from vertex s to t in graph"

E.g., driving directions, Internet routing.

Example:



Idea: given known distance to closest $S \subset V$, then distance of closest neighbor of S to s can be found. Then, induction.

Metric: shortest one-hop distance from vertices with known distances.

Update: (after processing vertex v)

- v's distance is known.
- update metric on unknown vertices if one-hop path from v is shorter.

Algorithm: Dijstra's Shortest Path Alg (w. Priority Q)

1. initialize
   (a) for all v, insert(v,∞)
   (b) deceasekey(v,0)
2. while queue not empty
   (a) (v,d) = deletemin()
   (b) if v = t, return d.
   (c) for each neighbor u of v:

   decreasekey(u,$d + c(v,u)$)

Runtime: $T(n,m) = m \log n$.

Correctness

Theorem: Dijkstra is optimal

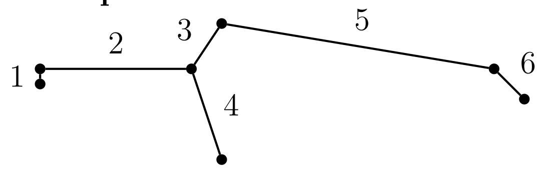
Proof: (by induction on known vertices, see text)

MSTs, revisited

Idea: grow tree from s by adding cheapest new vertex.

Note: as we add vertices, must reevaluate cost of vertices.

Example:



Idea: grow tree from start vertex adding closest vertex to any vertex in tree

Metric: minimum one-hop distance to any vertex in current tree.

Update: (after processing vertex v)

- add v to tree.
- update metric on non-tree vertices if one-hop distance to v is shorter.

Algorithm: Prim's MST Alg

1. initialize
   (a) for all v, insert(v,∞)
   (b) decreasekey(v,0)
2. while queue not empty
   (a) (v,d) = deletemin()
   (b) for each neighbor u of v:

   decreasekey(u,$c(v,u)$)

Runtime: $T(n,m) = O(n \log m)$

Correctness

Def: $A, B \subseteq V$ is a cut if $A \cup B = \emptyset$ and $A \cap B = E$. Edge $e = (u,v)$ crosses cut if $u \in A$ and $v \in B$ (or vice versa).

Lemma: (cut lemma) For any (A,B)-cut and $e' = (u,v)$ the min cost edge crossing cut, e' is in every MST.

Proof: (contradiction)

Conclusion: each edge Prim adds is minimum edge on cut, therefore Prim never adds wrong edge.