CS2100 - Shortest Path Algorithms

Note Title 11/28/2012

This week

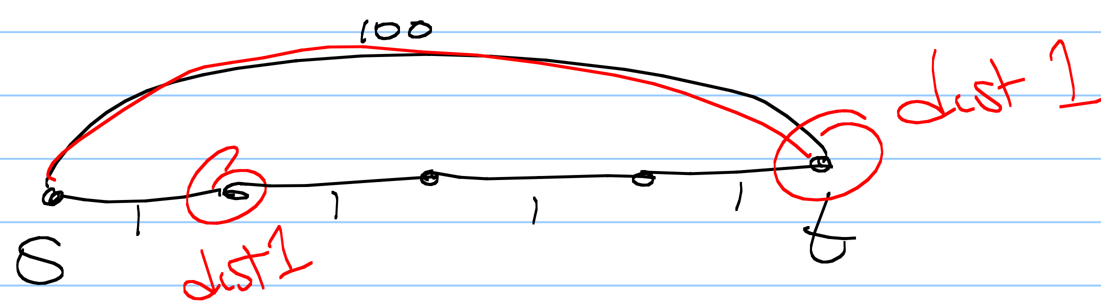
- No lab - lecture tomorrow
- Topics this week may be on exam!
- HW due in class on Friday
- Pass out sample finals on Friday
- Friday: review lecture, Monday review session
- Exam: next Wednesday at 8am

Other graph algorithms

- BFS returns a "short" s-t path, in some sense.

But won't work if graph has weights on the edges.

Why?



Which s-t path will be in BFS tree?
not shortest

Weighted Graphs

Formally, a weighted graph is a graph $G = (V, E)$ along with a function $w : E \to \mathbb{R}$ which gives each edge a weight $w(e)$.

The length of a path is the sum of the weights of the edges.

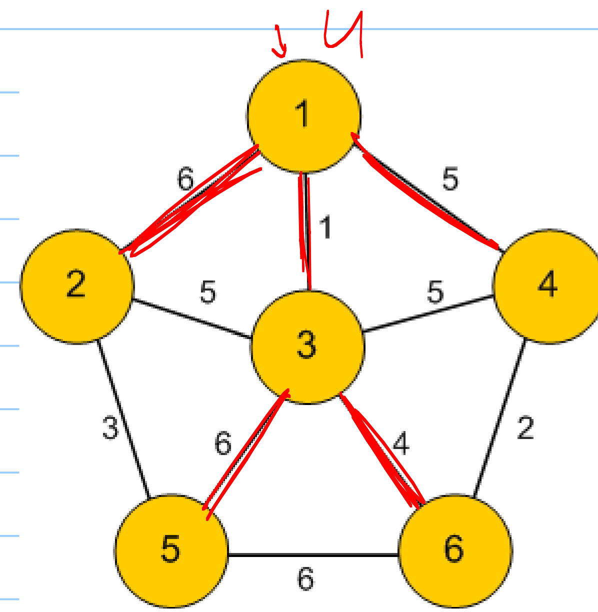
The distance $d(u,v)$ is the length of a minimum weight path.

Problem:

Given a weighted graph & a vertex u, compute the shortest paths from u to every other vertex.

How?

Simulate BFS, but watch weights



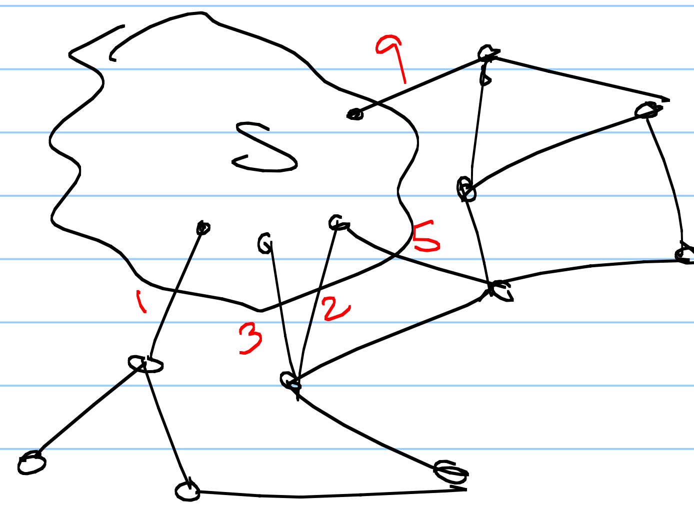
Dijkstra's Algorithm

Essentially, a "weighted" BFS.

We'll keep a set of vertices whose shortest path is known.

Want to add in the next closest vertex.

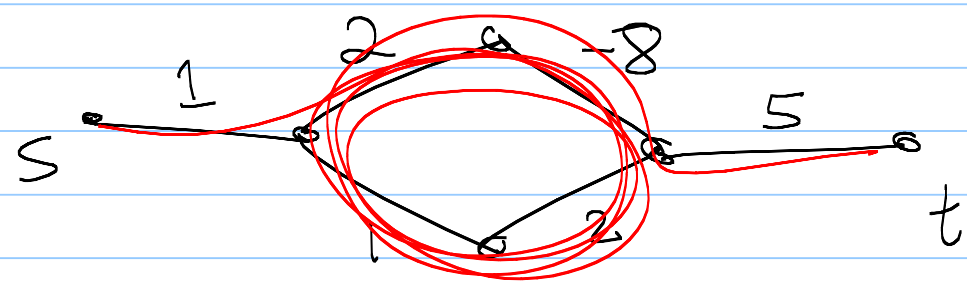
Key idea: This must give the shortest path to that vertex.



Details

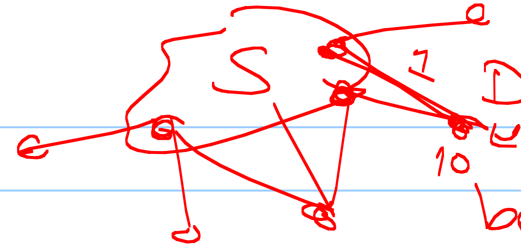
Assume G is undirected & has no negative edges.

Why?



What is the shortest path from s to t?

Edge relaxation



D[u] is current best distance

Key operation:

Suppose D[u] holds current best path from s to u (so far).

(Initially, D[s] = 0 & D[u] = ∞ for all u's.)

Need to update D label's via relaxation:

Say we update D[u].

For each edge (u, v),

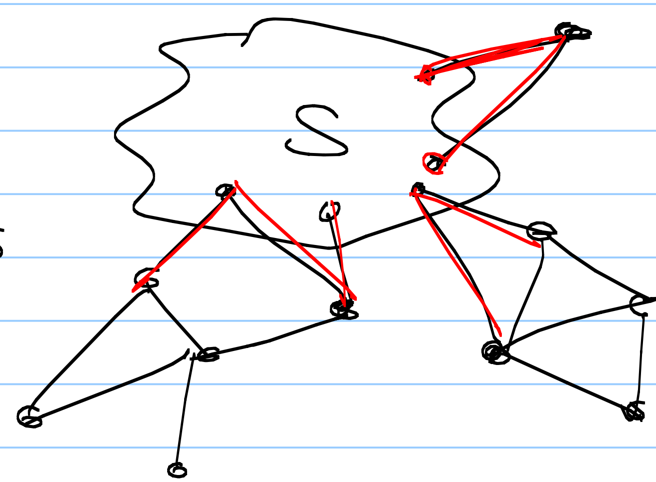
is D[u] > D[v] + w(u,v)?

if so, update D[u] so path goes through v

So at each phase, let S be set of "known" distances D[u].

Relax all neighbors of vertices in S.

Afterwards, can add the minimum of all the neighbors to S.

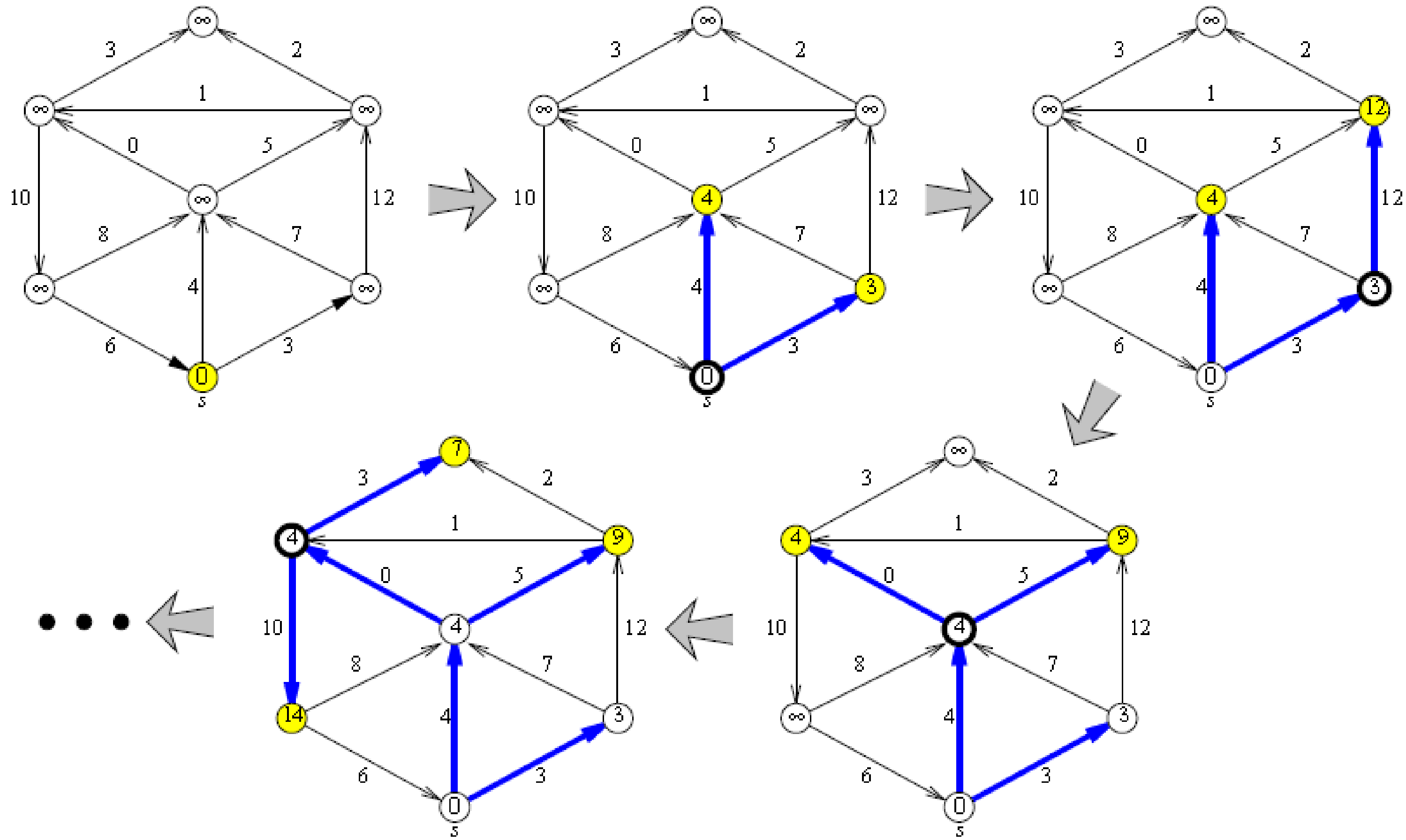


Algorithm: Given G & s:

Initialize $D[s] = 0$
For all $u \neq s$, set $D[u] = \infty$

Create a heap H & add $(D[s], s)$ to it

While H is not empty
    $u = H.\text{removeMin}()$
    for each neighbor v of u
        if $D[u] + w(u,v) < D[v]$
            $D[v] = D[u] + w(u,v)$
            Add new $(D[v], v]$ to Q



Four phases of Dijkstra's algorithm run on a graph with no negative edges.
At each phase, the shaded vertices are in the heap, and the bold vertex has just been scanned.
The bold edges describe the evolving shortest path tree.

Running Time

- removeMin: $O(\log n)$
  repeated at most n times - once per vertex
- relaxation of each adjacent edge: adding potentially $d(v)$ updates
  Note: Need an adaptable priority queue, where priorities can be updated in $O(\log n)$ time

Total: $\sum_{v \in V} (1 + d(v)) \log n$

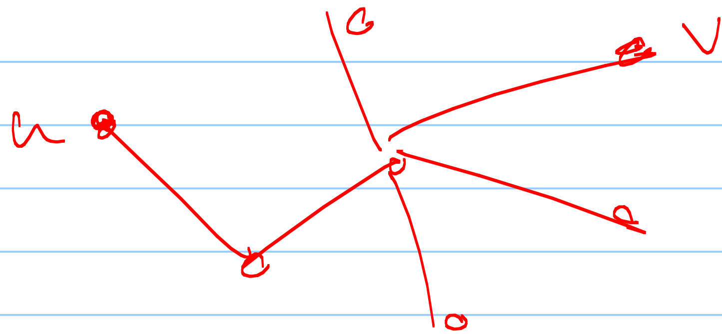
$$= \log n \left[\sum_{v \in V} (1 + d(v)) \right] = \log n \left[\sum_{v \in V} 1 + \sum_{v} d(v) \right]$$

$$= (m+n) \log n$$

Another question:

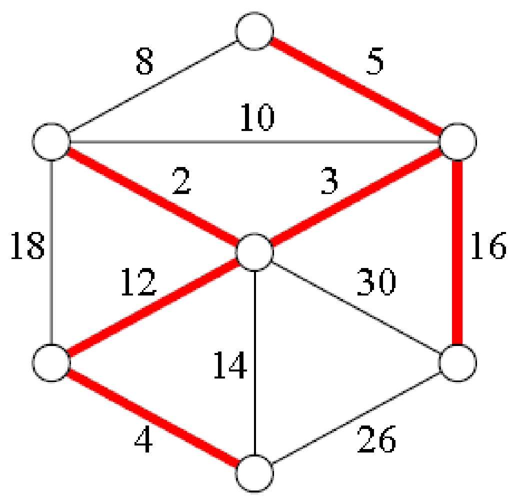
Given G, find a tree containing every vertex with minimum total weight.

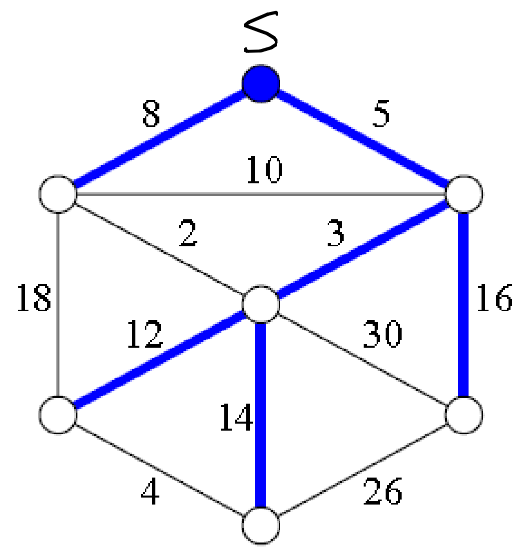
Uses? network design



This is called the minimum spanning tree of G.

Note: Not the same as shortest path tree!





Next time:

How to compute MST.