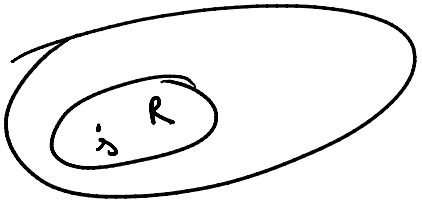
Shortest Paths

Thursday, September 24, 2015
4:40 AM

- Consider a directed, weighted graph G. For two given vertices s and t, what's the shortest path from s to t?
- Almost all algorithms for this problem actually solve the single source shortest path (SSSP) problem: Given source s, what are the shortest paths from s to every vertex in G.
- Observe: If shortest paths are unique, they form a tree. (Even if not unique, can choose them so that they form a tree)
- Easiest case: G is unweighted (all edge wts 1)
  - In this case, BFS works. $O(m+n)$ time

Dijkstra's algorithm

- Suppose edge wts are non-negative.
- Idea: Just like BFS, incrementally enlarge the set of vertices R for which distance from s is known. Want to put vertices in in order of distance from s.
- Initially, $R = \{s\}$ and $dist(s) = 0$.



Want to next insert $v \notin R$ that is closest to s. Note that in shortest path from s to v, the node before v must be in R, by positivity of edges (otherwise u is closer than v to s).

So, $dist(v) = dist(u) + d(u,v)$ for $u \in R$.

- So, Try all single-edge extensions of R, find the [illegible] extended path and put its endpoint in R. [illegible] edge.

- Need to inspect only edges touching new v

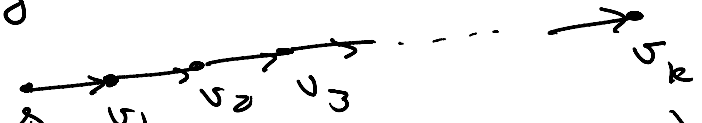
DIJKSTRA (G, s)
  dist $(s) = 0$
  dist $(v) = \infty$ for $v \neq s$
  $R = \{s\}$
  while $R \neq V$:
    Pick $v = \arg\min_{v \notin R}$ dist (v)
    Add v to R
    for $(v, w) \in E$:
      dist $(w) = \min($dist (w), dist $(v) + d(v, w))$

- Argue by induction: dist (v) is the shortest length of a path from s to v, all whose intermediate vertices are in R.
- If the dist values are kept in an array, complexity $O(n^2)$
- If in a min-heap, $O(m \log n)$

Bellman-Ford

- Consider directed graph with neg wt edges (we consider undirected later)
- Shortest path can pass thru vertices farther away.
- Neg wt cycles not allowed, as shortest paths undefined.
- Still, <u>relaxation</u> step OK:
  dist $(w) = \min($dist (w), dist $(v) + d(v, w))$

- Consider any shortest path



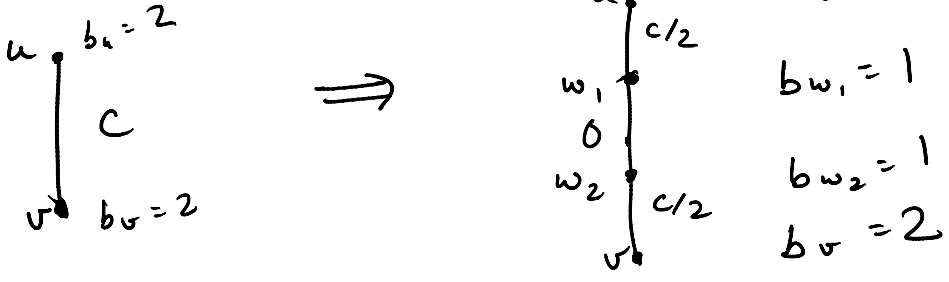
Note that if the edges $(s, v_1), (v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$ are relaxed in that order, then dist (v_k) is correct. Observe that it doesn't matter if there are other relaxations of other edges in between.

- But we don't know the right order. Simple but elegant solution: for $n-1$ times, relax all the edges (in any order)
- Time complexity : $O(mn)$
- To detect neg wt cycle, relax one more time to see if any dist value decreases.
  - If it decreases, then there is a shortest path of length n, meaning the path is not simple. Moreover, the cycle must have neg wt because otherwise there would be a shorter path with fewer edges
- Invariant after i iterations
  1) If dist $(v) < \infty$, it is the length of some path from s to v.
  2) dist (v) is at most the length of the shortest path from s to v with $\leq i$ edges.
- Application of neg wt cycle detection to finding arbitrages.

Shortest simple paths in undirected graphs

If negative, then can simply make each

- If all edge wts positive, then can simply make each edge two directed edges and run Dijkstra - But otherwise, quite complicated
- Approach: convert problem to that of finding min wt perfect matching in a graph.
- Latter problem solvable in time $O(mn + n^2 \log n)$ by Blossom algorithm (covered later, perhaps)
- Reduction:
  1. Add self loops to all vertices other than s and t.
  2. For an n-dimensional integer vector b, define b-matching to be collection of edges such that vertex i is incident to b_i edges (loops count twice)
  3. One can now try to split vertices v with $b_v = 2$. But then perfect matching could select the same original edge twice. To avoid this, we'll make sure that for any edge (u, v), either $b_u = 1$ or $b_v = 1$.



  4) Now split vertices v with $b_v = 2$ into two vertices and replicate incident edges on both

copies