Dijkstra's Algorithm

CS 62 - Spring 2016
Michael Bannister

This Week

Weekly Assignment

- 20 Questions Animal Game
- Binary trees
- Arbitrary length strings is **extra credit** and easy!

Weekly Lab

- Short lecture on makefiles
- Implement some graph algorithms

Reading: Bailey Chapter 16

Strong Connectivity

Question: How do we test if a graph is strongly connected?

Answer: (1) Run a search, without restarting from an arbitrary vertex *s* and see if all vertices get marked. (2) Reverse all of the edges in the graph. (3) Run a second search, without restarting, from *s* and see if all vertices are marked. The graph is strongly connected iff both searches mark all vertices

Runtime: O(n+m)

The Single Source Shortest Path Problem (SSSP)

Input: A graph with weighted edges and a start vertex *s.*

Output: The collection of shortest paths from *s* to every vertex in the graph reachable from *s*

SSSP Observation

- The collection of all shortest paths form a tree, called the shortest path tree
- If all edges have the same positive weight, then the problem is solved by BFS

SSP Algorithms

Dijkstra's Algorithm: Used when all of the edges have non-negative weights

Runtime: O((n+m) log n)

Bellman-Ford Algorithm: Used when negative edge weights are possible, but no negative cycles

Runtime: O(nm)

Dijkstra's Algorithm

Variables:

```
graph G                  //  The graph
vertex_t s               // The starting vertex
double length[n][n]      // Edge length
double dist[n]           // Current best distance
pqueue Q                 // PQ (vertex_t, double)
```

Dijkstra'a Algorithm

```
Set dist[v] to ∞ for all v, except dist[s] = 0
Add s to Q with priority 0.0
loop while Q is not empty
  get vertex cur with min priority d
  if d ≤ dist[curr]:
    for each out going edge cur → v:
      if dist[cur] + length[cur][v] < dist[v]:
        dist[v] = dist[cur] + length[cur][v]
        parent[v] = cur
        Add v to Q with new priority dist[v]
```

Dijkstra Example

(On Board)