The Shortest Path problem

- Given graph and a vertex $s$ find shortest paths from $s$ to all other vertices.
- Map routing, robot navigation, urban traffic planning
- Optimal pipelining of VLSI chip
- Routing of telecommunication messages
- Network routing protocols (OSPF, BGP, RIP)
- Seam carving, texture mapping, typesetting in TeX!
Example with positive edge weights
Example with negative edge weights
Unweighted shortest paths

- Given unweighted graph $G$
- Can assume all edge weights are 1
- Find shortest paths from $s$
- There is what is known as a shortest path tree!
- Can be found using Breadth First Search (BFS)
void Graph::unweighted( Vertex s ){
    Vertex v,w;
    s.dist = 0;
    for(int currDist=0; currDist < NUM_VERTICES; currDist++)
        for each vertex v
            if( !v.known && v.dist == currDist ){
                v.known = true;
                for each w adjacent to v
                    if( w.dist == INFINITY ){
                        w.dist = currDist + 1;
                        w.path = v;
                    }
            }
}
void Graph::unweighted( Vertex s ){

    Queue q( NUM_VERTICES );
    Vertex v,w;
    q.enqueue(s);
    s.dist = 0;

    while( !q.isEmpty() ){
        v = q.dequeue();
        v.known = true;
        for each w adjacent to v
            if( w.dist == INFINITY ){
                w.dist = v.dist + 1;
                w.path = v;
                q.enqueue( w );
            }
    }
}
Main structural properties of Shortest Paths

- Prefixes of shortest paths are themselves shortest paths
- Does a shortest path always exist?
- What about a shortest path tree?
- How can we compute such a tree
Main concepts

- known vertices
- Relaxation of an edge \((v, w)\): \(d(w) = \min(d(w), d(v) + c_{vw})\)
- Next: The Djikstra algorithm
“Do only what only you can do.”

“In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.”

“The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence.”

“It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration.”

“APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.”
Edsger W. Dijkstra: select quotes

"Object-oriented programming is an exceptionally bad idea which could only have originated in California."
-- Edsger Dijkstra
Dijkstra algorithm: arbitrary non-negative edge weights

- Store $d_v, known, p_v$
- Pick vertex with minimum $d_v$ (that is not known)
- Relax all edges outgoing from it
- Repeat until all vertices are known
Example of Dijkstra in action
Example of Djikstra in action
Example of Djikstra in action
Example of Dijkstra in action
Example of Djikstra in action
Example of Djikstra in action
Example of Djikstra in action
Example of Djikstra in action
Example of Djikstra in action
Dijkstra data-structures

struct Vertex
{
    List adj;   // Adjacency list
    bool known;
    DistType dist;
    Vertex path;  // ref to parent in path
};

void Graph::createTable( vector<Vertex> & t){
    readGraph( t ); //Read graph, fill in adj
    for(int i=0; i < t.size(); i++){
        t[i].known = false;
        t[i].dist = INFINITY;
        t[i].path = NOT_A_VERTEX;
    }
    NUM_VERTICES = t.size();
}
Shortest Paths after Djikstra run

```cpp
void Graph::printPath( Vertex v )
{
    if(v.path != NOT_A_VERTEX)
    {
        printPath( v.path );
        cout << " to ";
    }
    cout << v;
}
```
void Graph::dijkstra( Vertex s ){  
    Vertex v,w;
1.     s.dist = 0;
2.     for( ; ; ){  
3.         v = smallest unknown distance vertex;
4.         if( v == NOT_A VERTEX )
5.             break;
6.         v.known = true;
7.         for each w adjacent to v;
8.         if( !w.known )
9.             if( v.dist + c(v,w) < w.dist ){  
10.                 decrease w.dist to v.dist + c(v,w);
11.                 w.path = v;
12.             }
13.         }  
14.     }
15. }

The Djikstra algorithm: pseudo-code
Implementing Djikstra

- Naive implementation (using array to find min $d_v$) :
  $O(|E| + |V^2|) = O(|V|^2)$
- Could we be better for sparse graphs?
Implementing Djikstra

- Naive implementation (using array to find min $d_v$):
  $O(|E| + |V^2|) = O(|V|^2)$

- Could we be better for sparse graphs?

<table>
<thead>
<tr>
<th>PQ impl</th>
<th>insert</th>
<th>delete-min</th>
<th>decrease-key</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>unordered array</td>
<td>1</td>
<td>V</td>
<td>1</td>
<td>$V^2$</td>
</tr>
<tr>
<td>binary heap</td>
<td>log $V$</td>
<td>log $V$</td>
<td>log $V$</td>
<td>$E \log V$</td>
</tr>
<tr>
<td>$d$-way heap</td>
<td>$\log_d V$</td>
<td>$d \log_d V$</td>
<td>$\log_d V$</td>
<td>$\log \frac{E}{V} V$</td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>1</td>
<td>log $V$</td>
<td>1</td>
<td>$E + V \log V$</td>
</tr>
</tbody>
</table>
Negative edge weights!
Acyclic Graphs

- Important special case: Nonreversible chemical reactions, critical path analysis
- Running time is \( O(|E| + |V|) \)
- Dijkstra can be implemented along with Topological sort
Example: Activity-node graph
Event-node graph
Latest Completion times
EC, LC, Slack, Critical Path

Earliest and Latest Completion times

\[ EC_1 = 0 \]
\[ EC_w = \max_{(v,w) \in E} (EC_v + c_{v,w}) \]

\[ LC_n = EC_n \]
\[ LC_v = \min_{(v,w) \in E} (LC_w - c_{v,w}) \]

Slack of an edge \((v, w)\)

\[ Slack_{(v,w)} = LC_w - EC_v - c_{(v,w)} \]
The Bellman Ford algorithm

Basic Pseudo code

\[ d[s] = 0 \]
\[ \text{for } i = 1 \text{ to } |V| \]
\[ \quad \text{Relax each edge} \]

- Why does this work?
Bellman Ford: Queue Based Implementation

```cpp
void Graph::weightedNegative(Vertex s) {
    Queue q(NUM_VERTICES);
    Vertex v, w;
    q.enqueue(s);
    s.dist = 0;
    while (!q.isEmpty()) {
        v = q.dequeue();
        for each w adjacent to v
            if (v.dist + c(v, w) < w.dist) {
                w.dist = v.dist + c(v, w);
                w.path = v;
                if (w is not already in q)
                    q.enqueue(w);
            }
    }
}
```
Bellman Ford contd.

- Runtime is $O(EV)$
- Can be used to detect negative cycles
- Useful in finding arbitrage opportunities!
All-Pairs Shortest Path

- Can run $|V|$ Djikstra’s - $O(|E||V| \log |V|)$
- Floyd Warshall : Dynamic programming algorithm
- Works in $O(|V|^3)$