Weighted Graphs II

University of Massachusetts Amherst
ECE 242 – Data Structures and Algorithms
Lecture 27

Weighted graphs

• Edges have “weights”
  – Cost metric
• Example interpretations
  – Distance
  – Dollar cost of traversal
  – Delay
• Cost is assumed to be additive
Shortest path algorithm

• Find shortest path between two nodes
  – “distance” = “cost”
• Typical variant: find shortest path between one node and all other nodes
  – Shortest path tree from starting node

Shortest path example

• Example: least-cost path from BDL to any other airport

![Cost Matrix and Path Diagram]
Shortest path tree

• How to find shortest paths?

Ideas

• Easier to solve for vertices close to start
  – Neighbors are easy to determine
• Avoid adding edges where “future path” may be better
  – Cost is positive and additive
    (path cost can not go down later)
• Iteratively expand the set of nodes where shortest path is known
Dijkstra’s algorithm

• Assign starting vertex to set and set distance to zero
• Repeat until all nodes are in set
  — For all nodes j not in set
    • Calculate lowest distance to start vertex
      — Distance of node i in set + cost of edge from i to j
      — Remember predecessor node for j (i.e., i)
  — Pick node j not in set with lowest total distance
    • Add edge from predecessor to j to tree
    • Add j to set

Dijkstra’s algorithm

• Variables needed to maintain information
  — Set of nodes for which shortest path is set
  — Distance to start node (as far as known)
  — Predecessor node for current distance
    • Needed to reconstruct shortest path tree
Shortest path tree

- Try Dijkstra’s algorithm yourself from BDL!

Dijkstra’s algorithm example

- Initial setup: start node in set, distance 0
Dijkstra’s algorithm example

• Update distance to neighboring non-set nodes

Dijkstra’s algorithm example

• Pick non-set node with min distance: BOS $82
Dijkstra’s algorithm example

• Update distance to neighboring non-set nodes
  – Costs are additive

Dijkstra’s algorithm example

• Pick non-set node with min dist.: BTV $147
Dijkstra’s algorithm example

• Update distance to neighboring non-set nodes

Dijkstra’s algorithm example

• Pick non-set node with min dist.: EWR $167
Dijkstra’s algorithm example

• Update distance to neighboring non-set nodes
  – EWR: $327 \rightarrow $230

• Pick non-set node with min dist.: MHT $181
Dijkstra’s algorithm example

• Update distance to neighboring non-set nodes

Dijkstra’s algorithm example

• Pick non-set node with min dist.: LGA $230
  – Import. to remember predecessor (not shown)
Dijkstra’s algorithm example

• Update distance to neighboring non-set nodes

Dijkstra’s algorithm example

• Pick non-set node with min dist.: SWF $242
Dijkstra’s algorithm example

- Update distance to neighboring non-set nodes

Dijkstra’s algorithm example

- Pick non-set node with min dist.: PVD $291
Dijkstra’s algorithm example

• Update distance to neighboring non-set nodes
  – JFK finally gets cost

Dijkstra’s algorithm example

• Pick non-set node with min dist.: ALB $297
Dijkstra’s algorithm example

• Update distance to neighboring non-set nodes

• Pick non-set node with min dist.: HPN $307
Dijkstra’s algorithm example

- Update distance to neighboring non-set nodes

Dijkstra’s algorithm example

- Pick non-set node with min dist.: JFK $362
Dijkstra’s algorithm example

- Result:

![Graph showing Dijkstra's algorithm example]

Pseudo Code

```
1 Initialization:
2    N' = {u}
3    for all nodes v
4       if v adjacent to u
5          then D(v) = c(u,v)
6       else D(v) = ∞
7
8 Loop
9    find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N'
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15    until all nodes in N'
```
Code

• Set up of data structures:

```java
public Graph shortestPath(Vertex start) {
    Graph sp = new Graph(maxVertices);
    int[] distance = new int[activeVertices];
    final int infinity = Integer.MAX_VALUE;
    // define "infinity"
    boolean[] inSet = new boolean[activeVertices];
    Vertex[] predecessor = new Vertex[activeVertices];
    for (int i=0; i<activeVertices; i++) {
        // copy vertices to graph
        sp.addVertex(vertices[i]);
        distance[i] = infinity;
        inSet[i] = false;
        predecessor[i] = null;
    }
    ...
```  

• Adding of start node and updating of neighbor costs:

```java
distance[start.graphIndex]=0; // initialize starting vertex
inSet[start.graphIndex]=true; // make part of set
Vertex min=start; // min tracks last added node
for (int i=1; i<activeVertices; i++) {
    // run once for every non-start vertex
    for (int j=0; j<activeVertices; j++) {
        // update costs of min's neighbors
        if ((inSet[j] & edges[min.graphIndex][j]!=0) {
            // if edge to non-set vertex exists
            if (distance[min.graphIndex]+edges[min.graphIndex][j]<distance[j]) {
                // check if path via min is shorter
                distance[j]=distance[min.graphIndex]+edges[min.graphIndex][j];
                // if so, update distance
                predecessor[j]=min; // and predecessor
            }
        }
    }
    ...
```
Code

• Finding of min and adding of edge:

```java
min=null;
for (int j=0; j<activeVertices; j++) {
    // find non-set vertex with least distance
    if (!inSet[j] && (min==null || (min!=null && distance[j]<distance[min.graphIndex]))) {
        min=vertices[j];
    }
}
sp.addEdge(min, predecessor[min.graphIndex],
            edges[predecessor[min.graphIndex].graphIndex][min.graphIndex]);
// add edge with same weight as in original graph
inSet[min.graphIndex]=true; // mark vertex as part of set
```

Next Steps

• Lecture on Monday
• Project 4 due on Thursday 11/20!