CMSC 341
Lecture 21

Announcements

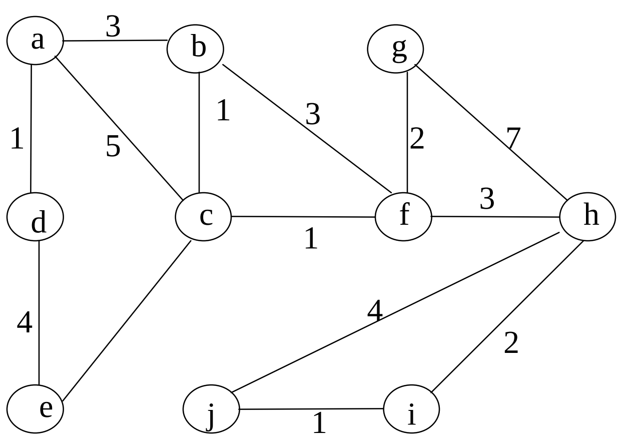
Clarifications on website for Proj5

Project Preview tonight and tomorrow

- Proj5 discussion
- dbx overview and tips

Dijkstra’s Algorithm

```
Vertex v, w;
start.dist = 0;
for (;;) {
   v = smallest unknown distance vertex;
   if (v == NOT_A_VERTEX) break;
   v.known = TRUE;
   for each w adjacent to v
        if (!w.known)
               if (v.dist + cvw < w.dist) {
                      decrease (w.dist to v.dist + cvw);
                      w.path = v;
                      }
   }
```



Edge Types

After DFS, edges can be classified into the following types:

- *tree edges* -- a discovered vertex v_1 encounters an undiscovered vertex v_2; the edge between them is a tree edge
- *back edges* -- a discovered edge v_1 encounters a discovered but unfinished vertex v_2; the edge between them is a back edge. (Graph has a cycle if and only if there is a back edge.)
- *forward edges* (directed graphs only) -- a discovered vertex v_1 encounters a finished vertex v_2
- *cross edges* (directed graphs only) -- a discovered vertex v_1 encounters a finished vertex v_2 and $d[v_1] > d[v_2]$

Edge Types (after DFS completion)

Condition	Type of Edge (v_1, v_2)
If $(d[v_1] < d[v_2])$ && $(f[v_1] > f[v_2])$	Tree
Else if $(d[v_1] > d[v_2])$ && $(f[v_1] < f[v_2])$	Back
Else if $(d[v_1] > d[v_2])$ && $(f[v_1] > f[v_2])$	Cross
Else $(d[v_1] < d[v_2]-1)$ && $(f[v_1] > f[v_2])$	Forward

Traversal Performance

What is the performance of DF and BF traversal?

Each vertex appears in the stack or queue exactly once. Therefore, the traversals are at least O(|V|). However, at each vertex, we must find the adjacent vertices. Therefore, df- and bf-traversal performance depends on the performance of the `getAdjacent` operation.

getAdjacent

Method 1: Look at every vertex (except u), asking "are you adjacent to u?"

```
List L = new List(<class of vertex>);
for (each vertex v, except u)
  if (isConnected(u,v))
   L.doInsert(v);
```

Assuming O(1) performance on `isConnected,` `getAdjacent` has O(|V|) performance and traversal performance is O(|V^2|);

getAdjacent (cont)

Method 2: Look only at the edges which impinge on u. Therefore, at each vertex, the number of vertices to be looked at is D(u), the degree of the vertex (use outdegree for directed graph).

This approach is O(D(u)). The traversal performance is

$$O(\sum_{i=1}^{|V|} D(v_i)) = O(E)$$

which is max(O(|V|), O(|E|))) = O(|V| + |E|).

Number of Edges

Theorem: The number of edges in an undirected graph G=(V,E) is $O(|V|^2)$

Proof: Suppose G is fully connected. Let p =|V|. We have the following situation:

vertex	connected to
1	2,3,4,5,…, p
2	1,3,4,5,…, p
…	
p	1,2,3,4,…,p-1

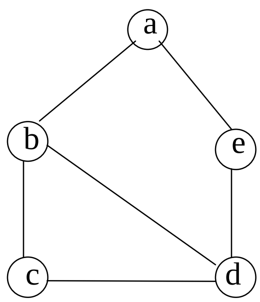
– There are p(p-1)/2 = $O(|V|^2)$ edges.

So O(|E|) = $O(|V|^2)$.

Adjacency Table Implementation

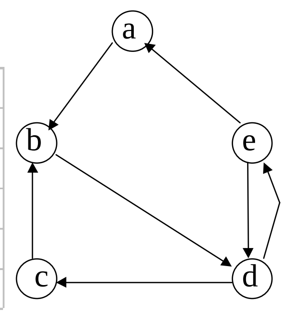
Uses table of size |V| × |V| where each entry (i,j) is boolean

- TRUE if there is an edge from vertex i to vertex j
- FALSE otherwise
- store weights when edges are weighted



	a	b	c	d	e
a	0	1	0	0	1
b	1	0	1	1	0
c	0	1	0	1	0
d	0	1	1	0	1
e	1	0	0	1	0

	a	b	c	d	e
a					
b					
c					
d					
e					



Adjacency Table (cont.)

Storage requirement:

Performance:

GetDegree(u), getInDegree(u), getOutDegree(u)		
GetAdjacent(u)		
GetAdjacentFrom(u)		
IsConnected(u,v)		