Program 5

Homework 9 due 10 pm Friday, April 22nd

Last Time
Graphs
• traversals
• applications of BFS/DFS
• more terminology

Today
Graphs
• topological ordering
• Dijkstra’s algorithm
Exam 2 returned

Next Time
Read: continue Hashing
Hashing
• terminology
• designing a good hash function
• choosing table size
• expanding a hash table
• handling collisions
Topological Ordering

1. get bread
2. get jelly
3. get peanut butter
4. get butter knife
5. open jelly
6. open peanut butter
7. take bread slice 1
8. take bread slice 2
9. use knife to spread jelly on bread slice
10. use knife to spread peanut butter on bread slice
11. put slices together with spreaded sides facing each other

IDEA:
Topological Ordering Algorithm

Iterative Algorithm (see readings for recursive algo)

Example
Dijkstra's Algorithm

Psuedo Code

for each vertex V
    initialize V’s visited mark to false
    initialize V’s total weight to “infinity”
    initialize V’s predecessor to null

set start vertex’s total weight to 0

create new priority queue pq
pq.insert( [start vertex total weight,start vertex] )

while !pq.isEmpty()
    [C’s total weight,C] = pq.removeMin()
    set C’s visited mark to true

    for each unvisited successor S adjacent to C
        if S's total weight can be reduced
            S's total weight = C's total weight + edge weight from C to S
            update S’s predecessor to C
            pq.insert( [S’s total weight,S] )
            (if S already in pq we’ll just update S's total weight)
Reconstruct shortest path from A to F

Dijkstra’s Practice

```
<table>
<thead>
<tr>
<th>Iteration</th>
<th>Priority Queue (just list smallest to largest)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Visited</th>
<th>Total Weight</th>
<th>Predecessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```

```