**COP 3503H– CS2 Fall 2011 Sample Midterm Questions**

1. D**epth first search** is the basis for many algorithms concerning graphs, both directed and undirected. Most of these algorithms start with a depth first search that provides a post-order numbering of all nodes.

 Assuming a graph with **n** nodes , **e** edges and **M=max(n,e)**, what is the complexity of assigning post-order numbers to the nodes of a graph?

 How may the post-order numbers be used to determine the existence of a cycle? In answering this, assume post[i] is the post-order number of node i and (i,j) denotes an arc (directed edge) from node i to node j.

 How can a depth first search determine if an undirected graph is connected?

**2.** The second implementations of Dijkstra’s shortest paths algorithm has an algorithmic structure that looks like

1. settled := [ FirstCity ]; unsettled = [ succ(FirstCity) .. LastCity ];
2. for v in unsettled do short[v] := dist[FirstCity,v];
3. structure short to suit our needs; // this may require short to be a pair (distance, node)
4. while unsettled <> [ ] do begin
5. find u in unsettled for which short[v] is shortest;
6. settled := settled + [u]; unsettled := unsettled - [u];
7. restructure short if necessary;
8. for v in unsettled do short[v] := min(short[v], short[u] + dist[u,v])
9. restructure short if necessary;
10. end;

 Assume **n** nodes (cities), **e** edges and **M=max(n,e)** where appropriate.

 **a)** What would be the cost of running this algorithmif we maintained **short** as a sorted list in ascending order? Explain how you arrived at your conclusion by indicating the contributions of specific numbered activities in the algorithm.

 **b)** What would be the cost of running this algorithmif we maintained **short** in a min heap data structure? Explain how you arrived at your conclusion by indicating the contributions of specific numbered activities in the algorithm.

**3.** Consider the scheduling problem where we have a set of dependent tasks running on a fixed number of processors, and we wish to minimize the time at which the last task completes.



 Now show an optimal (quickest time to complete all tasks) schedule. Be sure that your schedule obeys the precedence constraints given in the above graph.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 2 | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

 **4.** We described the proof that **3SAT** is polynomial reducible to Subset-Sum.

a.) Describe **Subset-Sum**

b.) Show that **Subset-Sum** is in **NP**

c.) Assuming a **3SAT** expression **(a + ~b + c) (b + b + ~c)**, fill in the upper right part of the reduction from **3SAT** to **Subset-Sum**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** | **a + ~b + c** | **b + b + ~c** |
| **a** | **1** |  |  |  |  |
| **~a** | **1** |  |  |  |  |
| **b** |  | **1** |  |  |  |
| **~b** |  | **1** |  |  |  |
| **c** |  |  | **1** |  |  |
| **~c** |  |  | **1** |  |  |
| **C1** |  |  |  | **1** |  |
| **C1’** |  |  |  | **1** |  |
| **C2** |  |  |  |  | **1** |
| **C2’** |  |  |  |  | **1** |
|  | **1** | **1** | **1** | **3** | **3** |

 **5.** Given that the frequency of letters in this sentence is
a – 1
c – 2
e – 7
f – 1
g – 1
h – 2
i – 4
l – 1
n – 5
o – 1
q – 1
r – 2
t – 7
u – 1
v – 1
y – 1
sp – 9

Create a Huffman code and specify how many bits it would take to encode the above sentence.

 **6.** Given the following weighted directed graph



 Compute the network flow from A to B. List each path that contributes to this flow, with all intermediate nodes, and the amount contributed.

 **7.** Given the following independent tasks and execution times
T1/7, T2/3, T3/5, T4/8, T5/2, T6/3, T7/4
Show a schedule on two processors that optimizes mean finishing times.
Show a schedule on two processors that minimizes the finishing time of the last task.
What is the complexity of each of these scheduling problems when dealing with an arbitray set of independent tasks?