Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Algorithms and Imperative Programming

Date: Thursday 20th May 2010
Time: 14.00 – 16.00

Please answer THREE Questions, of which at least one must be from Section A

Please use separate Answerbooks for EACH section

This is a CLOSED book examination

The use of electronic calculators is NOT permitted
Section A

1. Consider the following undirected weighted graph that represents a hypothetical network of airports. The weights of the edges represent the flight times between the airports.

   ![Graph Image]

   a) Present in the form of pseudocode Dijkstra’s algorithm for finding the shortest path between a given node and all other nodes. How is this used to find the shortest paths between all pairs of nodes?

   The algorithm requires a priority queue. You do not need to give pseudocode for the priority queue. You may assume the priority queue functions remove_smallest(), insert(int k) and change(int k) are already defined. These respectively remove the index with the smallest value, insert item with index k, and update the queue due to change in value of item k. (5 marks)

   b) Apply Dijkstra’s algorithm to the problem of finding the shortest path between the nodes B and G in the above graph. In other words, show the progression of the algorithm step by step. (5 marks)

   c) State the algorithmic complexity of Dijkstra’s algorithm for finding the shortest paths between all pairs of nodes and compare it to that of Floyd’s algorithm. Give a relation between the number of nodes and the number of edges in a graph for which the complexities of Dijkstra’s algorithm and Floyd’s algorithm become the same. (4 marks)

(Question 1 continues on the following page)
(Question 1 continues from the previous page)

d) Consider the heuristic path-following algorithm described in the lab. Explain what is meant by a heuristic. Under what circumstances would it be appropriate to use this approach? What are the disadvantages of the approach? (3 marks)

e) Name one heuristic you think would be effective for finding the shortest path between the two nodes of an airport graph. Justify your answer. Compare the paths found in part b) to the path found by the heuristic algorithm using your heuristic. (3 marks)
2. a) Explain clearly what is meant by a 0/1 Knapsack problem concerning a number of items each with a weight and a value. (3 marks)

b) Explain how a greedy algorithm may be used for the 0/1 Knapsack problem. Does the algorithm always give an optimal solution? (4 marks)

c) Consider the following 0/1 Knapsack problem with N=5 items and a Knapsack capacity (maximum weight) of C=10. The items have the following weights and values:

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Value</td>
<td>7</td>
<td>5</td>
<td>20</td>
<td>8</td>
<td>14</td>
</tr>
</tbody>
</table>

To solve this problem using dynamic programming, we generate a 2-D array \( A \) indexed by \( k \) (item numbers) and \( w \) (current maximum weight) using the following pseudocode (where \( w_k \) is the weight of the \( k \)-th item and \( v_k \) its value):

\[
\text{if } (w_k > w) \\
\quad v_{\text{new}} = A[k-1, w] //use previous row's value. \\
\text{else}
\quad v_{\text{previous_row}} = A[k-1, w] //use previous row's value. \\
\quad v_{\text{new}} = A[k-1, w-w_k]+v_k //the k-th item is used \\
\quad v_{\text{new}} = \max (v_{\text{previous_row}}, v_{\text{new}})
\]

\( A[k, w] = v_{\text{new}}. \)

Using this, we generate the following array \( A \) (with some missing values):

<table>
<thead>
<tr>
<th>( w )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0</td>
<td>7</td>
<td>8</td>
<td>21</td>
<td>22</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
</tbody>
</table>

(Question 2 continues on the following page)
(Question 2 continues from the previous page)

For example, we calculate $A[k, w]$ for $k=3$ and $w=10$. Here $v_k$ is 20 and $w_k$ is 9, so $w_k > w$ is false and so we enter the else clause.

Here
\[ v_{\text{previous\_row}} = A[2, 10] = 12 \]
\[ v_{\text{new}} = A[2, 1] + 20 = 7 + 20 = 27 \]

so
\[ A[3, 10] = \max(12, 27) = 27 \]

Complete the above table of values by calculating $A[5, 3]$ and $A[5, 10]$. Show your working. (5 marks)

d) What is the time complexity of this dynamic programming algorithm expressed in terms of the capacity $C$ and the number of items $N$? (2 marks)

e) Consider now the following 0/1 Knapsack problem with $N=5$ items and capacity $C=60$.

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>6</td>
<td>24</td>
<td>54</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>Value</td>
<td>7</td>
<td>5</td>
<td>20</td>
<td>8</td>
<td>14</td>
</tr>
</tbody>
</table>

What is the size of the table which is generated by the above dynamic programming algorithm? (2 marks)

f) Can this table size be reduced? Explain your answer. [Hint: consider the relationship of this Knapsack problem to the previous problem.] (2 marks)

g) Give a general rule which allows this form of reduction to take place for the dynamic programming solution of 0/1 Knapsack problems. (2 marks)
Section B

3. a) What is the worst-case time-complexity of EACH of the following general-purpose sorting algorithms?

- Heap Sort
- Bubble Sort
- QuickSort
- Selection Sort
- Merge Sort
- Insertion Sort

(3 marks)

b) Carefully explain the meaning of ANY THREE of the following terms related to sorting and sorting algorithms. Use examples where appropriate.

- in-place
- stable
- comparison-based sorting
- total order
- linear-time

(6 marks)

c) The following algorithm sorts an array of $N$ words, $A[ ]$, into dictionary order. It uses a bucket sort on each letter of the words in turn.

In the algorithm below: $A[i]$ is the $i$th word; $A[i,j]$ is the $j$th letter of the $i$th word; hash_value($A[i,j]$) returns the number 1 for '<no-letter>', 2 for 'a', 3 for 'b' ..., 27 for 'z'; there are 27 buckets, and each one is implemented as a (first-in-first-out) queue. Note: All words in the input array $A$ are in lower-case letters.

(Question 3 continues on the following page)
Radix_Sort_Words(A[ ], N)
1: M=0;    // the variable M will store the number of letters in the longest word
2: for (i = 1 to N)
3:   if (word_length(A[i]) > M)
4:      M = word_length(A[i]);
5:   for (j = M down to 1)  // main loop repeats M times
   // words are sorted by last letter first, then the next last, and so on
6:      for (i = 1 to N)  // loop through the words
7:         place_word_in_bucket( hash_value(A[i,j] ) );  // this operation takes O(1) time
8:      c=1;
9:   for (k=1 to 27)  // loop through the buckets
10:      while (bucket_not_empty(k))
11:         A[c++]=remove_next_item_from_bucket(k);  // this operation takes O(1) time
// end of main loop
//end

i) With reference to the pseudocode, derive the time complexity of the algorithm. SHOW YOUR WORKING. Your final answer should be in Big-O notation. (4 marks)

ii) What is the space complexity of the algorithm? SHOW YOUR REASONING. Your final answer should be in Big-O notation. (3 marks)

iii) Explain why the above algorithm sorts words correctly into dictionary order, i.e. “a” comes before “aardvark” comes before “abacus”, etc. (4 marks)
4. Consider the following heap, where the highlighted node containing 8 is the last element of the heap.

```
4
  
5
  15
  
9
  16
 25
14
12
11

6
  
7
  20
```

a) Assuming that a total order relation on the keys of a dataset is given (e.g. by a comparator), define the heap-order property of a binary tree. (4 marks)

b) Show the steps of inserting the key 5 into the above heap. (5 marks)

c) Give the time complexity of insertion into a heap. Explain your answer. (3 marks)

d) Show the steps of removing the element with the smallest key from the heap. (5 marks)

e) Give the time complexity of the removal operation from a heap. Explain your answer. (3 marks)
5. a) Explain the following two representations of directed graphs:
   i) Adjacency lists,
   ii) Adjacency matrices.

In your explanation you should define each representation and you should show how the following graph is represented as an adjacency list and as an adjacency matrix.

![Graph Diagram]

(3 marks)

b) Which of the two representations from Question 5(a) is most suitable for the following applications? Explain your answers, using time and space complexities where appropriate.
   i) Finding a node with maximum in-degree (the in-degree of node \( n \) is the number of edges whose target is node \( n \)),
   ii) Finding all nodes which are reachable by a path of length 2 from a given node,
   iii) Given a directed graph \( G \), the operation of constructing the opposite graph, which is the graph with the same nodes as \( G \) but reverses the direction of each edge of \( G \),
   iv) Representing a graph with many nodes but few edges. (6 marks)

c) How may the two above representations of graphs be used to represent undirected graphs? (2 marks)

d) Explain clearly what is meant by a depth-first search (DFS) of an undirected graph. You may give a program or a pseudocode description or a clear step-by-step explanation of the traversal method. (5 marks)

e) Explain how a DFS of an undirected graph may be used to compute the number of connected components of an undirected graph. You need not give a program or pseudocode, but should explain your method clearly. (4 marks)