UNIVERSITY OF MORATUWA

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BSc Engineering Honours Degree
Semester 2 Examination: 2011 Intake / Semester 4 Examination: 2010 Intake

CS2022: DATA STRUCTURES & ALGORITHMS

Time allowed: 2 Hours June 2013

ADDITIONAL MATERIAL: *None*

INSTRUCTIONS TO CANDIDATES:

1. This paper consists of **4** questions in **two (2) Sections** in **8** pages.
2. Answer all **4** questions. **Answer two sections in two separate booklets**.
3. Start answering each of the 4 main questions on a new page.
4. The maximum attainable mark for each question is given in brackets.
5. This examination accounts for 60% of the module assessment.
6. This is a closed book examination.

   NB: It is an offence to be in possession of unauthorised material during the examination.

7. Only calculators approved and labelled by the Faculty of Engineering are permitted.
8. Assume reasonable values for any data not given in or with the examination paper. Clearly state such assumptions made on the script.
9. In case of any doubt as to the interpretation of the wording of a question, make suitable assumptions and clearly state them on the script.
10. This paper should be answered only in English.

SECTION A

Q1. **[25 *marks*]**

a) Compare Quicksort, Merge sort and Heapsort considering their practical usage, worst case time complexity and space complexity.

[3 *marks*]

Let $A[1 \sim n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an ***inversion*** of A. (*Note: Pseudocodes for the Insertion sort, Merge Sort and Bubble sort algorithms are given at the end of the paper as resources.*)

b) List four (4) inversions of the array [5, 2, 8, 6, 1].

[2 *marks*]

c) What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions? How many does it have? Explain your answers.

[4 *marks*]

d) If the **actual number of inversions in the input array** is m, show that the running time of the Insertion sort will be $\Theta(m)$.

[3 *marks*]

e) What is the relationship between the **number of swaps that will occur in Bubble sort and the number of inversions in the input array**? Explain your answer.

[4 *marks*]

f)

i. Give an algorithm that determines the number of inversions in any permutation on n elements in $\Theta(n \lg n)$ worst-case time. (*Hint:* Modify Merge sort.)

[6 *marks*]

ii. Show that the worst case running time of the algorithm you proposed in part (e) is $\Theta(n \lg n)$.

[3 *marks*]

Continued …

Q2. **[25 *marks*]**

a) Briefly explain the two key properties an optimization problem should have if the dynamic programming approach is applicable to the problem. Using these properties explain why dynamic programming solutions would run faster compared to divide and conquer or recursive solutions.

[4 *marks*]

b) What is greedy choice property? Explain why greedy solutions would run faster compared to dynamic programming solutions.

[4 *marks*]

c) Rod cutting problem is defined as given a rod of length n meters and a table of prices p_i for i = 1, 2, …….., n (that represent the prices of pieces of length i for different i values), determine the maximum revenue r_n obtainable by cutting up the rod and selling the pieces.

i. Argue that the problem can be solved using the dynamic programming approach.

[2 *marks*]

ii. Derive the recursive solution to the problem of selecting the optimal set of pieces that will maximize the revenue for a rod of given length.

[3 *marks*]

iii. Given the following table with prices p_i for i = 1, 2, …….., 10, determine the maximum revenue obtainable by cutting up a 11 meter long rod using the dynamic programming approach. Clearly show the steps you followed.

length (i)	2	3	4	5	6	7	8
price (p_i)	5	8	9	10	17	17	20

[6 *marks*]

d) Consider a modified version of 1-0 knapsack problem in which the thief is allowed to **pick a part of only a single item.**

i. If **the item of which a part can be picked is fixed**, can the optimal solution be obtained using the greedy approach? Explain your answer.

[3 *marks*]

ii. If the **thief is given the flexibility to pick which item she wants to pick partially**, can the optimal solution be obtained using the greedy approach? Explain your answer.

[3 *marks*]

Continued …

SECTION B

Q3. **[25 *marks*]**

a) Consider the binary tree shown below (Figure Q3.a).

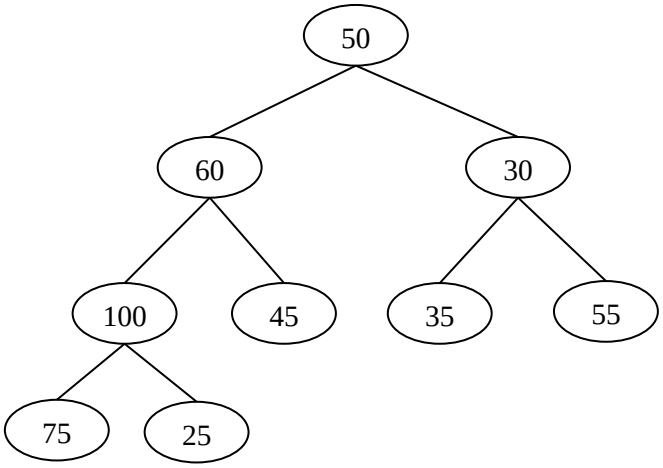

Figure Q3.a

i. Show how **Build-Max-Heap** operation will transform the above binary tree into a max heap. (You can show your steps using array representation or tree representation of heap.)

[4 *marks*]

ii. Show how Heapsort can be used to get the sorted list of numbers in the heap built in part (i). (You can show your steps using array representation or tree representation of heap.)

[4 *marks*]

b) Consider a connected weighted directed graph G = (V, E, w). Define the fatness of a path P to be the maximum weight of any edge in P. Explain an efficient algorithm that, given such a graph and two vertices u, v ∈ V , finds the minimum possible fatness of a path from u to v in G.
(Hint: You can modify one of the Single source shortest path algorithms we discussed in class to achieve this)

[5 *marks*]

c) In a treasure hunt game the treasure map is represented by figure Figure Q3.b. *P,Q,R,T,U* are treasures. Positive weights in edges represent the amount of energy you have to spend to traverse in each edge. Negative weights represent the energy boosts you can collect if you traversed through that edge.

Continued …

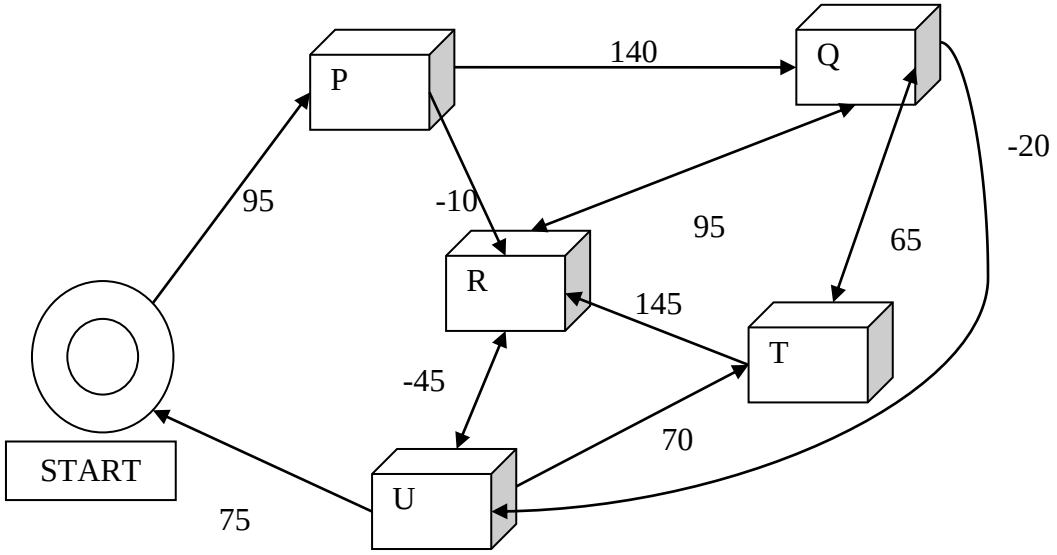

Figure Q3.a

i) You are starting the game at **Start position**. The game rules are such that once you have collected a treasure you should return to the starting position and then go for another treasure. You can only travel through the given edges in both arrival and return journey. If you want to **collect all the treasures** and **spend minimum amount of energy** for journey to collect treasures, **what are the paths you should take** to/from each treasure?

[7 marks]

ii) If you start with 1000 energy points will you be able to collect all treasures?

[2 *marks*]

iii) There is a guardian to protect treasures. Before you start the treasure the guardian wants to leave a message at each treasure and the starting point. Guardian also has to spend/ will gain the same amount of energy as shown the graph to send this message. How he can **leave this message at all treasures and starting point spending minimum amount of energy**?

[3 *marks*]

Q4. **[25 *marks*]**

a) Identify the most efficient data structures for following scenarios and justify your answer in one or two sentences

i) To implement a searching algorithm on a set of sorted numbers. The searching happens very frequently and the algorithm does random accesses on the data structure.

ii) To build a telephone directory to store telephone numbers of Colombo city residents. Search for a record should happen in nearly O(1) time.

Continued …

iii) To build a glossary to store nearly 300 different records. Search for a record should happen exactly O(1) time.

iv) To implement an algorithm to reverse any given string.

[2X4 *marks*]

b) .

[4 *marks*]

c) You are asked to implement two stacks in one array A[1…n] in such a way that neither stack overflows unless the total number of elements in both stacks together is n. Explain, using pseudo-code how PUSH (Stack, x), POP (Stack) operations are implemented in the given scenario. (The PUSH and POP operations should run in $O(1)$ time.)

[5 *marks*]

d) Consider the following set of values: 17, 9, 15, 16, 21, 10, 14, 25, 13, 7, 16, 20, 22

i) Insert above values in a Binary Search Tree (BST) in the given order.

[3 *marks*]

ii) Explain how you can sort the above values using the BST.

[2 *marks*]

iii) Explain using pseudo code how you can implement a recursive function to print all values in a BST less than a given key value.

[3 *marks*]

Continued …

Resources:

Pseudocode of Merge Sort:

```
MERGE-SORT(A, p, r)
1. IF p < r
2.    q ← ⌊(p + r)/2⌋
3.    MERGE-SORT(A, p, q)
4.    MERGE-SORT(A, q + 1, r)
5.    MERGE(A, p, q, r)

MERGE(A, p, q, r)
1.   n1 ← q - p + 1
2.   n2 ← r - q
3.   //create arrays L[1……n1 + 1] and R[1  n2 + 1]
4.   for i ← 1 to n1
5.      L[i] ← A[p + i - 1]
6.   for j ← 1 to n2
7.      R[j] ← A[q + j]
8.   L[n1 + 1] ← ∞
9.   R[n2 + 1] ← ∞
10.  i ← 1
11.  j ← 1
12.  for k ← p to r
13.     if L[i] ≤ R[j]
14.        A[k] ← L[i]
15.        i ← i + 1
16.     else
17.        A[k] ← R[j]
18.        j ← j + 1
```

Pseudocode of Insertion Sort:

```
INSERTION-SORT(A)
1.  for j = 2 to A.length
2.     key = A[j]
3.     //Insert A[j] into the sorted sequence A[1, ……, j-1].
4.     i = j-1
5.     while i > 0 and A[i] > key
6.        A[i+1] = A[i]
7.        i = i-1
8.      A[i+1] = key
```

Continued …

Pseudocode of Bubble Sort:

```
BUBBLE-SORT(A)
1. do
2.    swapped = false
3.    for i = 2 to A.length
4.       if A[i-1] > A[i]
5.          //Swap two elements
6.          temp = A[i]
7.          A[i] = A[i-1]
8.          A[i-1] = temp
9.          swapped = true
10. while swapped
```

--- End of the Paper ---

Continued …