ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
Department of Computer Science and Information Technology (CIT)

SEMESTER FINAL EXAMINATION **SUMMER SEMESTER, 2010 – 2011**
DURATION: 3 Hours **FULL MARKS: 150**

CIT 4405: Algorithms

Programmable calculators are not allowed. Do not write anything on the question paper.
There are **8 (eight)** questions. Answer any **6 (six)** of them.Figures in the right margin indicate marks.

---

1. a) Draw the recursion tree for the *Merge-Sort* procedure on an array of 8 elements. Explain why memoization fails to speed up a good divide-and-conquer algorithm such as *Merge-Sort*. 3+3

```
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)
```

   b) In the *longest-common-subsequence (LCS) problem*, we are given two sequences $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$, and we want to find a maximum-length common subsequence. Characterize optimal substructure of an LCS and provide a recursive solution, which are the first two steps to solve this problem using dynamic programming. 9

   c) Discuss the optimal substructure property that an optimization problem must have (along with overlapping subproblems propery) in order for dynamic programming to apply. Your discussion should include the following aspects: 10
      i. The pattern usually followed in order to discover optimal substructure.
      ii. Characterization of subproblem space.
      iii. The two ways in which optimal substructure varies across problem domains.

   Note: You should provide examples relevant to your discussion.

2. a) Draw the Huffman tree to get an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers: 10
   a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21
   Generalize your Huffman tree to find the optimal code when the frequencies are the first n Fibonacci numbers.
   Note: While generalizing the Huffman tree, represent each newly created node z as $z_1, z_2, z_3, \ldots, z_{n-1}$, each character c as $c_1, c_2, c_3, \ldots, c_n$, and each Fibonacci number F as $F_1, F_2, F_3, \ldots, F_n$.

   b) In activity selection problem consider any subproblem S_k, and let a_m be an activity in S_k with the earliest finish time. Prove that a_m is included in some maximum-size subset of mutually compatible activities of S_k. 8

   c) Justify the following statement: "fractional knapsack problem has the greedy-choice property". 7

3. a) Suppose that the algorithm *Connected-Components* is run on the undirected graph G = (V, E), where V = {a, b, c, d, e, f, g, h, i, j, k} and the edges E are processed in the order (d, i), (f, k), (g, i), (b, g), (a, h), (i, j), (d, k), (b, j), (d, f), (g, j), (a, e). List the vertices in each connected component after each iteration of lines 3-5. 10

```
Connected-Components (G)
1   for each vertex v ∈ G.V
2         Make-Set(v)
3   for each edge (u, v) ∈ G.E
4         if Find-Set(u) ≠ Find-Set(v)
5               Union(u, v)
```

b) i. In an attempt to provide faster implementation of disjoint sets, we represent sets by rooted trees, with each node containing one member and each tree representing one set, thus forming a disjoint-set forest. You know that two heuristics- "union by rank" and "path compression" can be used to improve the running time of the straightforward algorithm for disjoint-set forest. Discuss both of the heuristics. 10+5

ii. Which heuristic, among the two mentioned earlier, is used in the following procedure and how it works?

```
Find-Set (x)
1   if x ≠ x.p
2         x.p = Find-Set (x.p)
3   return x.p
```

4. a) i. Illustrate step by step the operation of *Max-Heapify (A, 3)* on the array A = {27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0}. 10+7

ii. Based on the procedure Max-Heapify (provided below), write pseudocode for the procedure *Min-Heapify (A, i)*, which performs the corresponding manipulation on a min-heap.

```
Max-Heapify (A, i)
1    l = LEFT (i)
2    r = RIGHT (i)
3    if  l ≤ A.heap-size and A[l] > A[i]
4          largest = l
5    else  largest = i
6    if  r ≤ A.heap-size and A[r] > A[largest]
7          largest = r
8    if  largest ≠ i
9          exchange A[i] with A[largest]
10         Max-Heapify (A, largest)
```

b) Briefly describe the operations supported by a *max-priority-queue*. 8

5. a) i. Show step by step execution of *Prim's algorithm* on the graph provided in Figure 1. 15+5

ii. Provide a simple analysis on the running time of *Prim's algorithm*.

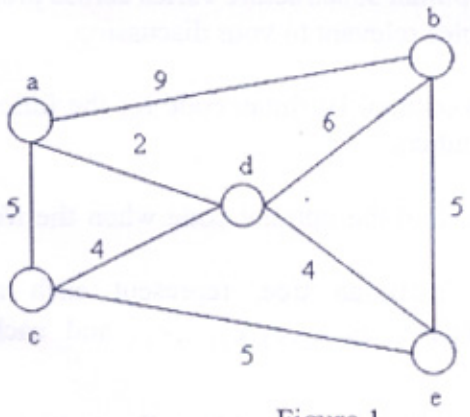

Figure 1

```
Mst-Prim (G, w, r)
1    for each u ∈ G.V
2          u.key = ∞
3          u.∏ = NIL
4    r.key = 0
5    Q = G.V
6    while Q ≠ ∅
7          u = Extract-Min (Q)
8          for each v ∈ G.Adj[u]
9               if v ∈ Q and w(u, v) < v.key
10                   u.∏ = u
11                   v.key = w(u, v)
```

b) Define the term *light edge* along with an example. 5

6. a) i. Show step by step execution of *Dijkstra's algorithm* on the graph provided in Figure 2 below. (node *a* is the source *s*) 10+5

ii. Provide a simple analysis on the running time of *Dijkstra's algorithm* provided below.

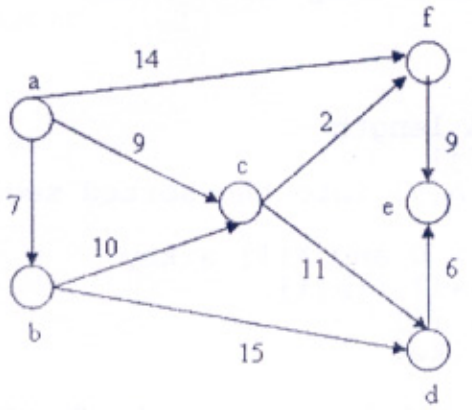

Figure 2

```
Algorithm SSSP-Dijkstra (G, w, s)
1   Initialize-Single-Source (G, s)
2   S = ∅
3   Q = G.V
4   while Q ≠ ∅
5         u = Extract-Min (Q)
6         S = S ∪ {u}
7         for each vertex v ∈ G.Adj[u]
8             Relax (u, v, w)
```

b) Prove the following Lemma: 10

Let G = (V, E) be a weighted, directed graph with source s and weight function $w: E \to R$, and assume that G contains no negative-weight cycles that are reachable form s. Then after the |V-1| iterations of the *for* loop of Bellman-Ford (where each iteration relaxes all the edges of the graph once), we have v.d = δ (s, v) for all vertices v that are reachable from s.

7. a) i. Write down the recurrence equation which describes the running time of Algorithm Power(x, n) as a function of n. 2+10

ii. Solve the recurrence using substitution method. As you already know that in substitution method you need to guess the solution at first before proving it, your task is to guess the solution of the recurrence using recursion tree and later prove it using substitution method (for only asymptotic upper bound).

```
Algorithm Power(x, n)
Input: A number x and integer n ≥ 0
Output: The value x^n
1   if n == 0
2         return 1
3   if n is odd
4         y = Power(x, (n-1)/2)
5         return x*y*y
6   else
7         y = Power(x, n/2)
8         return y*y
```

b) Using *Master method* we found that the solution of the recurrence $T(n) = 8T(n/2) + cn^2$ is $O(n^3)$. Show that a substitution proof with the assumption $T(n) \le d(n^3)$ fails. Then show how to subtract off a lower-order term to make a substitution proof work. 13

8. a) Observe that the while loop of lines 5-7 of the *Insertion-Sort* procedure uses a linear search to scan backward through the sorted subarray A[1…*j*-1]. Can we use binary search instead to improve the overall worst-case running time of *Insertion-Sort* to $\Theta(n \lg n)$? Justify your answer. 7

```
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
```

b) The algorithm *Memoized-Cut-Rod-Aux* provided below for solving rod-cutting problem does not take into account the cost of a cut. As nothing in this world is free, consider that for each cut there is a constant cost c (e.g., c=2). Now modify the algorithm in order to reflect the effect when considering the cost of a cut. 10

```
Memoized-Cut-Rod (p, n)
1   let r[0...n] be a new array
2   for i = 0 to n
3         r[i] = -∞
4   return Memoized-Cut-Rod-Aux(p,n,r)

Memoized-Cut-Rod-Aux (p, n, r)
1   if  r[n] ≥ 0
2         return r[n]
3   if  n == 0
4         q = 0
5   else  q = -∞
6         for i = 1 to n
7           q = max(q,p[i]+Memoized-Cut-Rod-Aux(p,n-i,r)
8   r[n] = q
9   return q
```

c) Use the definition of Big-Oh to prove the following: 4+4

i. $f(n) = .001n^3 - 1000n\lg n - 100n + 5n$ is $O(n^3)$

ii. $f(n) = (n+5)\log_2(3n^2+7)$ is $O(n\log_2 n)$