# ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

**ORGANISATION OF ISLAMIC COOPERATION (OIC)**

**Department of Computer Science and Engineering (CSE)**

**MID SEMESTER EXAMINATION** **SUMMER SEMESTER, 2018-2019**

**DURATION: 1 Hour 30 Minutes** **FULL MARKS: 75**

## CSE 4403: Algorithms

**Programmable calculators are not allowed. Do not write anything on the question paper.**

There are **4 (four)** questions. Answer any **3 (three)** of them.

Figures in the right margin indicate marks.

---

1. a) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, prove that $f(n) = O(h(n))$. 5

   b) A polygon is convex if all of its internal angles are less than $180°$ (and none of the edges cross each other). We represent a convex polygon as an array $V[1 \ldots n]$ where each element of the array represents a vertex of the polygon in the form of a coordinate pair $(x, y)$. $V[1]$ is the vertex with minimum $x$ coordinate and the vertices $V[1 \ldots n]$ are ordered counterclockwise. Assume that the $x$ coordinates of the vertices are all distinct, as are the $y$ coordinates of the vertices. 10×2

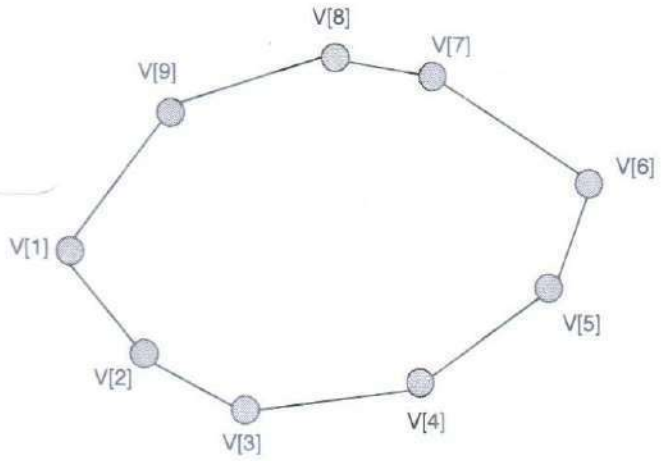


Figure 1: An example convex polygon with $n = 9$ vertices

   i. Give an algorithm to find the vertex with the maximum $x$ coordinate in $O(\log_2(n))$ time.

   ii. Give an algorithm to find the vertex with the maximum $y$ coordinate in $O(\log_2(n))$ time.

2. a) Let $A[1 \ldots n]$ be an array with $n$ distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair $(i, j)$ is called an inversion of $A$. Here, $1 \leq i, j \leq n$. 6+4

   i. What array with elements from the set $\{1, 2, \ldots, n-1, n\}$ has the most inversions? How many inversions does it have?

   ii. Is there any relation between the running time of insertion sort and the number of inversions in an input array? Justify your answer.

   b) In an effort to make merge sort faster, you decide to divide the array into $k$ equal sized, disjoint subarrays, where $k > 2$. You have to merge $k$ lists which can be accomplished in $O(n \log_2(k))$ time (Assume). Write the recurrence relation for your algorithm and find the running time of the algorithm. 5

   c) Argue why it is impossible to find a comparison-based sorting algorithm that sorts 5 numbers using at most 6 comparisons in the worst case. 5

   d) Suppose a binary max-heap contains 80 distinct keys. How many distinct positions might contain the smallest element? In which level from the top will they reside in? 5

3. a) You have a hash table with $m$ slots, with simple uniform hashing assumption. Collisions are resolved by chaining. What is the probability that the first slot ends up empty after $n$ insertions? 6

   b) In a hash table where collisions are resolved using chaining, we can replace the linked list with balanced BSTs. Give one reason why it might be useful. Give one reason why it might not be useful. 4+4

   c) Consider using a hash table with $m = 11$ slots for integer values and collision resolving using open addressing. Our hash function is: $h(k, i) = (f(k) + i \times g(k))\%m$, where $f(k) = k\%m$ and $g(k) = (k^2 + 1)\%m$. The table is initially empty. We perform the following operations sequentially: 6+5

   - Insert 3
   - Insert 14
   - Insert 90
   - Insert 2
   - Delete 14

   Now answer the following questions:

   i. Show the contents of the table with values and flags.

   ii. If we search for 13, what is the sequence of slots that we check?

4. a) Give reasonably simplified equation using Newton's Method to find $\sqrt[5]{a}$. Will the algorithm always correctly converge regardless of the choice of the initial guess $x_0$? 6+6

   b) You came up with a modified version of Karatsuba Multiplication where you divide the number in 3 parts. You reduced the total number of multiplications to 6. Is your algorithm better than that of Karatsuba? 6

   c) While calculating the complexity of High Precision Division $\left(\frac{a}{b}\right)$, we only considered the complexity of calculating $\left(\frac{R}{b}\right)$ ignoring the fact that we still need to find $\left(a \times \frac{R}{b}\right)$ which requires High Precision Multiplication. How does that affect the running time of Newton's Method for finding square roots? 7