United International University
CSI 227: Algorithms, Summer 2018
Mid-Term Exam
Total Marks: 90, Time: 1 hour 45 minutes

Answer all questions

1. (a) Derive the running-time equations for the function in Fig. 1 and express in $\mathcal{O}$ notation. [5]

   (b) For the function in Fig. 2 provide both **best** and **worst** case examples of A and B with size, len(A)=8. Also, derive the **worst** case run-time and express in $\mathcal{O}$ notation. [3+3+4]

   (c) Using the **recursion tree method**, determine a good asymptotic upper bound on the following recurrence: $T(n) = 4T(\frac{n}{3}) + n^2$. Show details of your calculation. [15]

```
1  Algorithm1(n):
2  sum = 0
3  for i=1 to n/2:
4      for j=1 to 100:
5          sum += i*j
6          ++j
7      ++i
8  return sum
```

Figure 1: Q. 1(a)

```
1  Algorithm2(A, B):
2  m = A.length
3  n = B.length
4  for i=1 to m:
5      for j=1 to n:
6          if (A[i] - B[j] <= 0):
7              break
8          j*= 3
```

Figure 2: Q. 1(b)

2. (a) Propose a **divide-and-conquer** algorithm to find the count of prime digits in an input integer n. Your algorithm should divide a problem of size m to 2 sub-problems of size $\frac{m}{2}$. What are the time-complexities of each of the divide, conquer and combine steps? [9+3]

   (b) Given an array $A = \{-2, 1, 0, -5, 3, -4, 7, -5\}$, find the **minimum sum continuous sub-array** using **divide-and-conquer** approach. You must show the recursion tree and clearly mention left, right and crossing sum for each tree node. [8]

3. (a) Provide an example where the greedy strategy fails for 0/1 knapsack problem for $n = 4$ items with the max-capacity of $W = 60kg$. [5]

   (b) Given the arrival and the departure times of n trains for a railway platform, provide a greedy algorithm to find out the maximum number of trains that can use that platform without any collision. Note that, there must exist at least 10 minutes of safety break between the departure of one train and arrival of a next one. [10]

4. (a) A Dynamic Programming algorithm for the classic Coin Change problem is provided at Fig. 3 where given an amount n and m types of coins C_1, C_2, ... , C_m, you have to minimize the number of coins to build up the amount n. Now, suppose that, each coin type C_i has a weight W_i. Modify the algorithm such that now you have to minimize the total weight of coins to build the amount n. [10]

```
let M[0..n] be the memoization table initialized to -1
let C[0..m] be the coins array
CoinChange(n):
if n = 0:
    return 0
if M[n] == -1:
    minval = +INF
    for i=1 to m:
        if C[i] <= n:
            tval = 1 + CoinChange(n-C[i])
            minval = min(tval, minval)
    M[n] = minval
return M[n]
```

Figure 3: Q. 4(b)

(b) Assume that in the context of classic **0/1 knapsack problem**, the capacity of the knapsack is $C = 5$. The store has 5 different products of values $V = \{8, 7, 5, 4, 9\}$ with weights $W = \{2, 2, 1, 1, 2\}$. Using **dynamic programming** solve the problem for this input set to find the maximum value a thief can acquire. You must show the corresponding recursion-tree and the memoization table produced for your solution. [15]