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

Answer all the questions from section A, and any 3 out of the 5 in section B. ($6 \times 15 = 90$).

Section A

1. (a) Derive the **best** and **worst** case running-time equations for the function in Fig. 1 and express in $\mathcal{O}$ notation. [8]

   (b) What are the 2 properties that make a problem a good candidate for the greedy algorithm? [7]

```
Algo(A,B):
n = A.length;
m = B.length;

for(i=0; i<n; i++)
        for(j=0; j<m; j++)
                if(A[i] == A[j])
                        return i,j;
return -1,-1;
```

Figure 1: Q. 1(a)

```
test(A)
        k = length(A);
        while (k > 1) {
                sum += A[k];
                if (sqrt(sum) == 5) return;
                k = k/2;
```

Figure 2: Q. 5(a)

2. Propose a **divide-and-conquer** algorithm to find the count of even numbers in an input array. Then using the **recursion tree method**, determine a good asymptotic upper bound on your solution. [5+10]

3. Harry was bored and hungry sitting idly in Hogwarts express. He requires exactly e more units of energy to reach Hogwarts. Fortunately, Hermione's knapsack contains m different types of potions of unlimited supply, serving energy of $E_1, E_2, \ldots, E_m$ units. Harry does not want to seem greedy so he decided to pick the **minimum number** of potions to serve his requirement of e energy units. Propose a **dynamic programming** algorithm to solve his problem. [15]

Section B

4. (a) Provide the recurrence relation for the **divide-and-conquer** function in Fig. 3 in terms of input array (A) size n, where $n = h - l + 1$. [5]

```
fun(A, l, h):
    if(l<h):
        sub = (h-l+1)/4
        fun(A, l, l+sub-1)
        fun(A, l+ 3*sub, h )

        for(i =l; i<=h; i++):
            printf(A[i])
```

Figure 3: Q. 3(a)

(b) What is the optimal-substructure of the **activity selection** problem? Briefly explain with an example. [5]

(c) Provide separate examples with exactly 4 activities where the greedy algorithm outputs sub-optimal solution if a greedy choice is made by: (i) shortest interval, (ii) earliest start time. [5]

5. (a) For the function in Fig. 2 provide both **best** and **worst** case examples of A with size, n=8. [5]

(b) Assume that in the context of classic **0/1 knapsack problem**, the capacity of the knapsack is $C = 4$. The store has 4 different products of values $V = \{8, 7, 5, 4\}$ with weights $W = \{2, 2, 1, 1\}$. Populate the memoization table for this input set to find the maximum value a thief can acquire. [10]

6. (a) Given an array $A = \{3, -4, 2, -3, -1, 7, -5\}$, 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. [10]

(b) For the function in Fig. 4 derive the **exact cost equation** and express in $\mathcal{O}$ notation. [5]

```
avada_kedavra(m, n):
    for(i=1;i<=m;i*=2):
        print i
    for(i=1;i<=n;i++):
        for(j=1;j<=100;++j):
            print i*j
```

Figure 4: Q. 6(b)

```
expecto_patronum(n):
    if (n==0):
        return 0
    minval = +INF
    for(i=0;i<n;++i):
        minval = min(minval,
            expecto_patronum(n-i-1))
    return minval
```

Figure 5: Q. 8(a)

7. (a) For **activity selection problem** we choose the earliest finished activity as our greedy choice. Prove that with this greedy choice, activity selection problem shows both "Greedy Choice Property" and "Optimal Substructure Property". [7]

(b) What is the fundamental difference between **divide-and-conquer** and **dynamic programming**? [5]

(c) What are the resources that have trade-off relations in dynamic programming? [3]

8. (a) Add **memoization** the function in Fig. 5 to get its **dynamic programming** version. Then demonstrate the recursion tree for calculation of expecto_patronum(3). [8]

(b) Using the **substitution method**, determine a good asymptotic upper bound on the following recurrence: $T(n) = T(n-1) + n$. Show details of your calculation. [7]