**(CS1.301) Algorithm Analysis and Design (Monsoon 2022)** Date: 22.09.2022

# Mid-semester examination

**Alloted time: 90 minutes** **Total marks: 25**

**Instructions:**

- There are a total of 5 questions over 2 pages with varying credit.
- No clarifications shall be provided during the examination. In case of doubts, please make a reasonable assumption and that could be considered for grading.
- Discussions amongst the students are not allowed. No electronic devices nor notes/books of any kind are allowed.
- Any dishonesty shall be penalized heavily.
- Be clear in your arguments. Vague arguments shall not be given any credit.

**Question 1** [3 marks]

Let G be a simple graph on $n$ nodes, where $n$ is an even number. If every node of G has degree at least $n/2$, then G is connected.

**Question 2** [1+2+1 marks]

1. Compute third primitive root of unity.
2. Construct the DFT and inverse DFT matrices of order $3 \times 3$.
3. Compute the DFT of the vector $(1, 1, 1)$ (without simplifying the expressions).

**Question 3** [4 marks]

**Algorithm 1:** OneDimensionalPeak(Array $A[1, n]$)

```
1  if length(A) ⩽ 3 then
2  |  return a peak here by brute-force.
3  end
4  middle ← ⌈length(A)/2⌉;
5  if A[middle − 1] ⩽ A[middle] and A[middle] ⩽ A[middle + 1] then
6  |  return middle;
7  else
8  |  if A[middle − 1] ⩾ A[middle] then
9  |  |  return OneDimensionalPeak(Array A[1, middle − 1]);
10 |  end
11 |  return OneDimensionalPeak(Array A[middle + 1, n]);
12 end
```

Given an array A with $n$ elements, we would like to find an index $i$ of a peak element $A[i]$ where $A[i] \geq A[i-1]$ and $A[i] \geq A[i+1]$. For elements on the boundaries of the array, the element only needs to be greater than or equal to its lone neighbor to be considered a peak.

Argue the correctness of Algorithm 1 for finding a peak, and compute its running time. In particular, argue that every given array will contain a peak. Here the notation $A[i, j]$ corresponds to the sublist of elements starting from index $i$ until the index $j$.

## Question 4

[6 marks]

Suppose you are a winner of a Golden ticket and you are invited to a huge chocolate factory. There you are shown $n$ different chocolates where the $i$'th item is worth $v_i$ rupees, and a total quantity of $w_i$ kilograms is available for taking. You are then handed a bag with a capacity of $W$ kilograms, and are asked to pick any item as long as your bag was not full to its capacity. What would you do if you want to maximize the value of contents in your bag. [Assume that any fractional quantity of each chocolate can be taken, and $w_i$'s are much smaller than $W$.]

## Question 5

[8 marks]

Given a sequence $a_1, \ldots, a_n$, we say that two indices $i < j$ form an inversion if $a_i > a_j$, that is, if the two elements $a_i$ and $a_j$ are "out of order". We seek to determine the number of inversions in the sequence $a_1, \ldots, a_n$.

The basic idea is to divide the list into the two pieces $a_1, \ldots, a_m$ and $a_{m+1}, \ldots, a_n$. We first count the number of inversions in each of these two halves separately. Then we count the number of inversions $(a_i, a_j)$, where the two numbers belong to different halves. We would like to do this merging part in $O(n)$ time.

Please read through the following code and fill in the missing snippets (in Algorithm 3) so that merging can be done in $O(n)$ time. Also construct the recursive relation for time complexity and state the correct running complexity.

**Algorithm 2:** Sort-and-Count(L)

```
if the list L has one element then
    return "There are no inversions";
A ← first ⌈n/2⌉ elements of L;
B ← the remaining ⌊n/2⌋ elements of L;
(r_A, A) = Sort-and-Count(A);
(r_B, B) = Sort-and-Count(B);
(r, L) = Merge-and-Count(A, B);
return r = r_A + r_B + r, and the sorted list L;
```

**Algorithm 3:** Merge-and-Count(A,B)

```
Maintain a Current pointer into each list, initialized to point to the front elements;
Maintain a variable Count for the number of inversions, initialized to 0;
while both the lists A and B are nonempty do
    TODO: FILL THE MISSING CODE
return Count and the merged list;
```