CENG 315 ALGORITHMS
MIDTERM EXAM FALL 2020

Written answer ODTUClass upload deadline: 2.12.2020 - 19:40
Oral exam: 2.12.2020 – until 20:30 (tentative)

- You are going to write your handwritten solutions to a **single blank paper**.
- You have to write **your name and your id** number at the top of the page.
- For each question, write only what is asked. Otherwise your answer will not be evaluated.
- You will take the **picture of your paper and upload as a single jpg** file. (PDF submission is not recommended but will be accepted.
- You have to write in a **clear and readable** form. Otherwise your answer will not be evaluated.
- **Any form of collaboration is strictly forbidden**.
- Any form of communication (except with the exam proctors and instructors in exceptional cases) is strictly forbidden throughout the exam duration.
- You cannot use any resource during exam hours (books, slides, etc. are all forbidden).
- There will be randomized **oral exams** during/after the completion of the exam. In case of a discrepancy between the written answers and the accuracy in answering related oral questions, disciplinary action may be taken.
- Until the exam (written **and** oral exam) completion is announced, the exam continues and you are required to stay online.
- The questions are confidential and cannot be shared, **you are required to delete any copies of the exam you have upon the completion of the exam**.
- Any violation of the rules may result in zero grade for the exam and/or disciplinary investigation.
- **Entering the exam effectively implies pledging that you will comply with all of the rules.**

Your answer paper should look like this:

```
NAME: ALİ ÖRNEK NUMBER: 123456
Q1)   T(n) = …
Q2 A) T(n) = …
Q2 B) k = …
Q2 C) T(n) = …
Q3 A) T(…) = …
      T(…) = …
      T(…) = …
Q3 B) T(n) = …
Q3 C) T(n) = …
      T(n) = …
      T(n) = …
Q4 A) T(n) = Σ …
Q4 B) T(n) = …
Q5 A) T(…,…)
Q5 B) T(i,j) = …
Q5 C) θ(...)
Q6 A) f(t,n) = …
Q6 B) …
```

QUESTION 1 (10 pts): Solve the following recurrence relation by using the *master theorem* method (using the given definitions).

T(1)=1
T(n) =4T(n/4)+n lgn

ONLY WRITE THE COMPLEXITY

QUESTION 2 (20 pts): Solve the following recurrence relation (for simplicity), defined **for powers of 4 only,** by using the *iterative* (top down) method. That is, through iterations, replace the recursive term k times. Then, determine k in terms of n, and write the solution.

T(1)=1
T(n) =4T(n/4)+n

PART A) ITERATIONS – ONLY WRITE T(n) AFTER k STEPS (don't write intermediate steps)

PART B) DETERMINE k – ONLY WRITE k IN TERMS OF n

PART C) DETERMINE T(n) – ONLY WRITE T(n)

QUESTION 3 (20 pts): Solve the following recurrence relation by using the *substitution* (bottom-up) method only **for powers of 2 values of n** (Hint: guess the solution and then prove by induction).

T(1)=1
T(n)=4T(n/2)+1

PART A) WRITE ONLY 3 STEPS OF BOTTOM UP SOLUTION

PART B) WRITE ONLY THE GUESSED SOLUTION

PART C) INDUCTION PROOF (write only 3 lines: recurrence relation/replacement/result) Skip the details from replacing the recursive term to the results

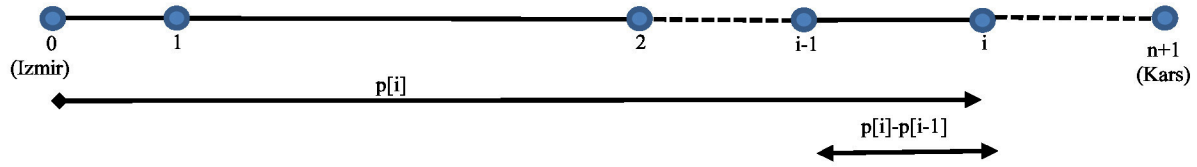
QUESTION 4 (20 pts): Determine the expression that gives the value of the variable **count** as a function of **n** (which describes the complexity of the function **f** in terms of **n**). Simplify your result containing at most 2 terms.

```
void f( int n)
{    int i,j,k; int count=0;
     for(i=1;i<=n;i++)
     {    j=n;
          while (j>=1)
          {    for (k=1;k<=j;k++)
                    count++;
               j=j/2;
          }
          }
}
```

PART A) WRITE ONLY THE SUMMATION FORM CORRESPONDING TO THE LOOPS

PART B) WRITE ONLY THE FINAL FORM OF THE CLOSED FORM OF THE SUMMATION

QUESTION 5 (20 pts):



Suppose you are the bus driver of a tourism company in Turkey. Your task is to drive a bus filled with tourists from the Izmir to Kars, by visiting a predefined list of cities, over exactly ***k* days**. You are required to drive only forward towards Kars (you cannot go back and re-visit a city).

There are $\boldsymbol{n}$ cities between Izmir and Kars. Each city is represented by an integer index from 0 to $n + 1$, as in the figure. The array $d[0, \dots, (n+1)]$ represents the distance of each city from Izmir, where $d[i]$ is the distance to **i**-th city in km's. By definition, $d[0] = 0$.

The company own a hotel at each city, and, you have to spend each night in a city. Let $\boldsymbol{v_j}$ be the **index of the city that you stay on day *j***. By definition, $v_0 = 0$ (Izmir) and $v_k = n + 1$ (Kars).

You need to choose the city to spend each night based on two factors:

1. Some hotels are nicer than the others, the company wants you to spend the night at the nicer ones as much as possible. As an incentive, the company pays you differently for spending the night at each city, given by $r[i]$ for city i.
2. You are also expected to drive around μ km's on each day. For this reason, there is a penalty for driving less than or more than μ km's, defined as the squared difference between the amount that you drive and μ. The penalty for day j is therefore given $\left(d[v_j] - d[v_{j-1}] - \mu\right)^2$.

The total amount of money that you will receive at the end of the trip is defined by:

$$\sum_{j=1}^{n+1} \left\{ r[v_j] - \left(d[v_j] - d[v_{j-1}] - \mu\right)^2 \right\}.$$

Manage your driving speed such that you get a maximum payment. Develop a dynamic programming approach, where $T(i,j)$ is the maximum payment you can receive for the days 0…j if you spend the night at city i on day j.

Answer the three following questions. Each answer can be expressed in just one line.

(a) [5 pts] Using $T(i,j)$ notation, express the value of the maximum payment for the whole trip.

(b) [10 pts] Write a recurrence that expresses the value of $T(i,j)$ for each $i \in \{1, \dots, n+1\}$ and $j \in \{1, \dots, k\}$. (ie. you need not worry about the base case $i = 0$)

(c) [5 pts] If you convert the recurrence into an efficient bottom-up algorithm, what will be its asymptotic complexity in θ notation?

QUESTION 6 (10 pts):

Remember the Take Home Exam 5 from last week:

> You are asked to implement a decision making mechanism for stock buying from the financial market. Your aim is to find the investment that generates the best outcome for you.
>
> For this, our AI module generated *(<time x stocks>)* 2-d array *market* where each *market[i][j]* shows the future outcome of stock *[j]* if bought at time *[i]*. However, there are 3 rules in this market:
>
> 1. You must buy only one *(exactly one)* stock at a given time (discrete time steps)
> 2. You cannot buy a stock twice.
> 3. You cannot buy a stock at a given time if a *bigger id* stock is already been bought in earlier time steps. (e.g. if you choose to buy stock 5 at time *t=2*, you cannot consider buying stocks 0,1,2,3 or 4 after that point, i.e. for *t = 3,4 ...*)
>
> Your task is to generate a stock buying order sequence to maximize your outcome.

Now suppose that we add a 4th rule:

4. **No** consecutive stocks (in terms of stock ids) can be bought. (e.g. you cannot buy stock 5 at time t=2 and then stock 6 at $t \geq 3$).

Answer the following two questions.

a) [7 pts] Write a recurrence that expresses the optimal value for buying stock n, at time t for $t \geq 1$ as a function f(t,n) (you do not need to consider the base case t=0):

b) [3 pts] Express the final optimal value in terms of f(t,n) for an input with T days and N stocks:
