Gebze Technical University Department of Computer Engineering CSE 321 Introduction to Algorithm Design Fall 2020

Midterm Exam (Take-Home) November 25th 2020-November 29th 2020

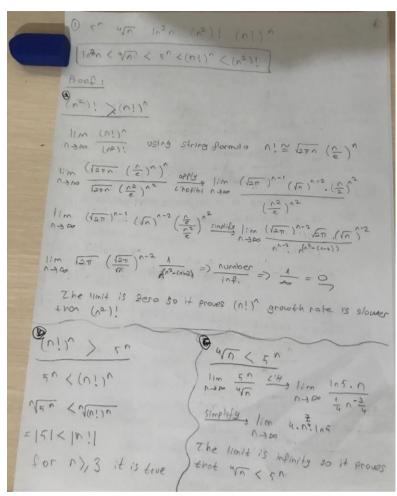
	Q1 (20)	Q2 (20)	Q3 (20)	Q4 (20)	Q5 (20)	Total
Student ID and						
Name						
OZAN YEŞİLLER						
171044033						

Read the instructions below carefully

- You need to submit your exam paper to Moodle by November 29th, 2020 at 23:55 pm <u>as a single PDF</u> file.
- You can submit your paper in any form you like. You may opt to use separate papers for your solutions. If this is the case, then you need to merge the exam paper I submitted and your solutions to a single PDF file such that the exam paper I have given appears first. Your Python codes should be in a separate file. Submit everything as a single zip file.
- **Q1.** List the following functions according to their order of growth from the lowest to the highest. Prove the accuracy of your ordering. **(20 points)**

Note: Your analysis must be rigorous and precise. Merely stating the ordering without providing any mathematical analysis will not be graded!

- a) 5ⁿ
- b) ∜n
- c) $ln^3(n)$
- d) $(n^2)!$
- e) $(n!)^n$



In
$$\frac{4\pi}{n \to \infty}$$
 $\frac{4\pi}{1.3n}$ $\frac{4\pi}{24 \cdot n \to \infty}$ $\frac{4\pi}{31 \cdot n \to \infty}$ $\frac{4\pi}{31 \cdot n \to \infty}$ $\frac{4\pi}{121 \cdot n^{2}}$ $\frac{4\pi}{n \to \infty}$ $\frac{4\pi}{n$

Q2. Consider an array consisting of integers from 0 to n; however, one integer is absent. Binary representation is used for the array elements; that is, one operation is insufficient to access a particular integer and merely a particular bit of a particular array element can be accessed at any given time and this access can be done in constant time. Propose a linear time algorithm that finds the absent element of the array in this setting. Rigorously show your pseudocode and analysis together with explanations. Do not use actual code in your pseudocode but present your actual code as a separate Python program. **(20 points)**

```
FUNCTION binary (arr, bitimen, bitsize = 32)
         index = bit Index - (bit size - length of orr))
         if arr[Index-1] == "0"
          then do return o
        else if orr [index-1] == " ["
         then do return 1
          throw exception ( "involid or ay")
FUNCTION find Missing (arr, bit Pos, LHSize)
     if 64 Pos < 0
       then do return o
     oddArr -> emptyList
    evenAm - emptylist
    index -0
  -bound - length of array
  while (index < bound)
      if binary (arrCindex], bit Pos, bit Size) == 0
        then do even Arr. append (arr [index])
     else
      then do oddArr. append (arr [index])
     index - index + f
   if length of odder >= length of eventure
    then do return find Missing (even Arr, bit Pos-1, bitsize) << 110
  else
   then do return findulssing (oddArr, bit Pos-1, bit Size) << 111
```

2. Complexity Analysis

This algorithm contains a recursive call. At any step of recursion the algorithm iterate on array almost half of length of array at each step. So the algorithm iterates are

According to master theorem

$$T(n) = \alpha \cdot T(\frac{n}{5}) + f(n)$$

d - 1, according to this algorithm \ while a < 6 = c - 1, according to this algorithm \ then a(n)

$$T(n) = T\left(\frac{\Lambda}{2}\right) + n$$

time complexity => o(n)

Q3. Propose a sorting algorithm based on quicksort but this time improve its efficiency by using insertion sort where appropriate. Express your algorithm using pseudocode and analyze its expected running time. In addition, implement your algorithm using Python. (**20 points**)

```
3 FUNCTION parsetur (arr. 1, n)
      PIU - arr [m]
       1+7+1
      for i=0 to range(1,4)
        if arraid L Piv
            suapping acid and acid
           うつうすけ
     Swapping act] and ach]
     F muter
  FUNCTION quicks (or, 1, h)
     if(1< h)
       then do
          Piu = Passe Arr (or, 1, h)
         quicks (arr, 1, Piv+1)
         quicks (arr, pivel, h)
     return arr
 FUNCTION insertions (or, 1: h)
     for i=0 to range (1+1, n+1)
       value = arr [i]
        while 3) low and arr [5-1] > val
           [1-72770 = [72770
           5-1 5-1
       arr [7] = val
     return or
```

FUNCTION hybrid (arr, 1, 11)

while Ich do

if h-1+1 > 15 Piu = porsextr(arr,1,h)

if piu-1) h-piu

then do

hybrid (arr, piu+1, h)

h >> piu+1

else

then do

hybrid (arr, 1, piu-1)

1 -> piu+1

else then do insertions (arr, 1, h)

Complexity Analysis and Explanation

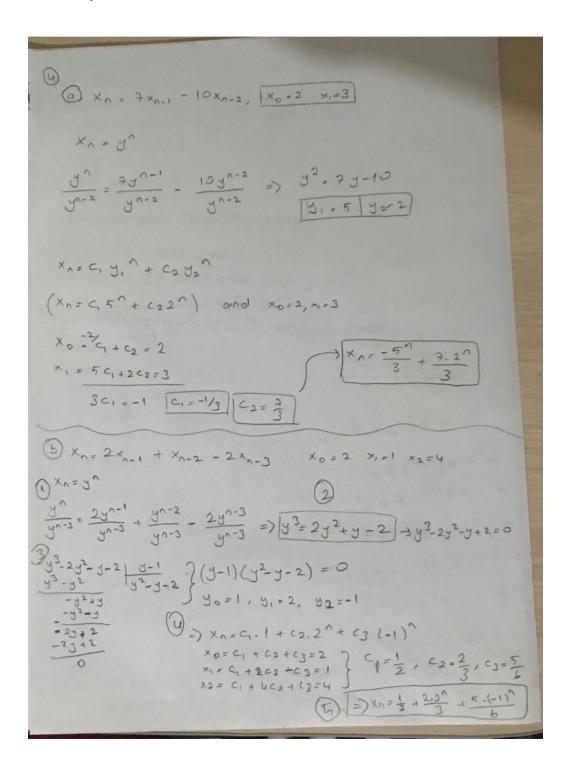
If the length of array is very large then quick sort is very efficient. But insertion sort more efficient than quick sort if the array length is very small. Number of comparisions and swap. Ping are less compared to quick sort. I combined this algorithms if the number of array size is less than 15 using insertion sort otherwise using quick sort.

quick sort runs average achlogal, when array's 12e is small, insertion more efficient than quick sort.

Therefore this algorithm time complexity is & (n 1091)

Q4. Solve the following recurrence relations

- a) $x_n = 7x_{n-1}-10x_{n-2}, x_0=2, x_1=3$ (4 points)
- b) $x_n = 2x_{n-1} + x_{n-2} 2x_{n-3}, x_0 = 2, x_1 = 1, x_2 = 4$ (4 points)
- c) $x_n = x_{n-1}+2^n$, $x_0=5$ (4 points)
- d) Suppose that a^n and b^n are both solutions to a recurrence relation of the form $x_n = \alpha x_{n-1} + \beta x_{n-2}$. Prove that for any constants c and d, $ca^n + db^n$ is also a solution to the same recurrence relation. (8 points)



Q5. A group of people and a group of jobs is given as input. Any person can be assigned any job and a certain cost value is associated with this assignment, for instance depending on the duration of time that the pertinent person finishes the pertinent job. This cost hinges upon the person-job assignment. Propose a polynomial-time algorithm that assigns exactly one person to each job such that the maximum cost among the assignments (not the total cost!) is minimized. Describe your algorithm using pseudocode and implement it using Python. Analyze the best case, worst case, and average-case performance of the running time of your algorithm. (**20 points**)

```
arr 84 S1 2e = 4
   init (self):
      self. porent = None
      Selp +otal Cost = 0
     self. cost=0
    self. worker = 0
    self. 10b=0
    Self. assigned = [Folse] & array Size
   def - It - (self. other):
       return self cost < other. cost
FUNCTION NEWDES (31, 52, assigned, parent)
     06=065()
     05. assigned = assigned
     05. ossigned [ 52] = True
     05. porent = porent
     ob. worker= 11
     06. 706 = 52
   return on
```

```
FUNCTION findlin Coot (or):
    queve = Priority aveve
    assigned - [false] & amoy size
    ( oot = new 0 65 (-1, -1, ossigned, 065 () )
    root. Fotal cost = root. cost = 0
    root, worker = -1
   queve. Put (root)
   while (not queue. empty ()):
       min = queve, get()
       i = min. worker +1
       if ( i = = arroy size):
          return mincost
    for 7 in rage (0, array stree):
       if (not minn.assigned[s]):
          child = newobs (1,5, min . assisped, minn)
          child total cost = minn. total cost + or (i) []
          child.cost = child.total Cost + cost Cake (argi, 5,
                                                 child.osraned)
```

```
FUNCTION cost Cole (ar, 71, 52, ospismed):
    cost = 2
   available = [True] = argyrae
   7= 71+1
   while(i < orgy size):
     Ethnishi= nim
      min_ind = -1
     J=0
     while () carroysite):
        if (not assigned [7] and out lable [7] and are is in in!
           min-ind= 5
            (15(:) 20 = nim
        3+=1
     cost += min
     available [min-ind] = false
    1+=1
  return cost
```

J wrote the Branch and Bound algorithm for this problem
in python loguege.

Zhis algorithm runs O(N4)

covered

while (19. emptyc)):

for 5 to N

furction call

O(N) × O(N) × O(N) × O(N)

- O(N4)