MSE | MASTER OF SCIENCE IN ENGINEERING

Final Exam Autumn Semester 2022/23

Modul / *Module*: **TFP Alg**
Datum / *Date*: **07.02.2023**
Dozierende/r / *Lecturer/s*: **Canci Jung Kyu**
Teil / *Part*: 1 von 2 / *1 of 2*

Name / Last name: ---

Vorname / First name: ---

E-Mail-Adresse / Email address: ---

FH / UAS: ---

Teilschule / Part of UAS: ---

Max. Punktzahl/max. points: 40
Erreichte Punktzahl/ Schlussnote/
Reached points / final mark: ---

Allgemeine Hinweise

1. Es werden während der Prüfung keine Fragen zu den Aufgaben beantwortet. Ist Ihnen eine Frage unklar, dann treffen Sie eine Annahme und erklären Sie diese in Ihrem Lösungsweg. Sie wird bei der Korrektur berücksichtigt.
2. Kommunizieren während der Prüfung ist grundsätzlich verboten. Mobiltelefone sind auszuschalten.
3. Open Book (auch in digitalen Form). Internet ist nur erlaubt, um die Prüfung zu herunter- und hochladen.

Viel Erfolg!

General information

1. *No question concerning the problems will be answered during the exam. If you don't understand a problem, make an assumption and explain it in your solution. It will be considered by the grader.*
2. *Communication with others during the exam is forbidden. Mobile phones must be turned off.*
3. *Open Book (also in digital form). Internet is just allowed to download and upload the exam*

Good luck!

Exercise 1 (4P)
Write in increasing order from left to right the following list of Theta classes. When two classes are equals, make it clear.
$\Theta(n^3)$, $\Theta(\log(n^2))$, $\Theta(n\log(n))$, $\Theta(n^2\log(n))$, $\Theta(\sqrt{n})$, $\Theta(n!)$, $\Theta(n^n)$,$\Theta(\cos(n))$, $\Theta(10^6)$

Answer

$\Theta(10^6)= \Theta(\cos(n))<\Theta(\log(n^2)<\Theta(n\log(n))<\Theta(n^2\log(n))<\Theta(\sqrt{n})<\Theta(n^3)<\Theta(n!)<\Theta(n^n)$

0.5 points each correct ordering symbol.

Exercise 2 (4P)
Write a pseudcode, which takes an array with n numbers and checks, whether there are at least two elements of the array that are equal.
Give the big O class of the running time of the pseudocode.

A possible answer. Input an array $A=[a_1,\ldots,a_n]$

```
DUPLICATES(A)
        found = false
        for i from 1 to n-1
                for j=i+1 to n compares a[i] and a[j]
                if a[i]=a[j] found=true
                else leave found unchanged
        return found
```

The comparison between two entries a[i], a[j] costs constant time c. We have to check n-1, n-2, …, 1 pair of entries. Therefore the running time is

$c[(n-1)+(n-2)+\ldots+1]=c*n*(n-1)/2$

which is $O(n^2)$.

2P for pseudocode 2P for running time

Exercise 3 (6P)
Consider the following pseudocode that takes as input a positive integer n and as output the number k, which is the biggest non-negative integer such that $2^k \leq n$ (the k-th power of two must be less or equal to n).

```
---------------------------------------------------------------------------------------------------
POWEROFTWO(n)
        If n≤1
          return 0
        else
          return 1+ POWEROFTWO(n/2)
---------------------------------------------------------------------------------------------------
```

Determine the recursive formula for determining the running time T(n).
If possible calculate explicitly the big O class of T(n) by using Master Theorem.

Answer :

We have
T(1)=c , constant time
$T(n)=T(n/2)+\Theta(1)$

Therefore we are in the case 2. of Master Theorem with a=1 and b=2, since the function f(n) in Master Theorem is $\Theta(n^{\log_2 0})= \Theta(1)$. Thus $T(n)= \Theta(\log n)$.

2P for correct formula for T(n) 4P for application of Master Theorem.

Exercise 4 (6P)
Consider the following recursive algorithm which, given an array a containing n integers, calculates the sum of the values contained in the array in the positions from the -lth and the r-th (both included).

```
Sum(a,l,r)
        if l=r
           return a[l];
        else
           centre =⌊(l + r)/2⌋;
           sum1 = Sum(a,l,c);
           sum2 = Sum(a,c+1,r);
        return sum1+sum2;
```

Determine the recursive formula for determining the running time T(n), in the case l=1 and r=n.
If possible calculate explicitly the big O class of T(n) by using Master Theorem.

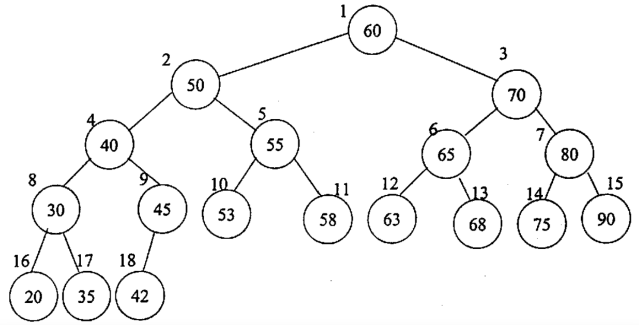
Answer:

We have
T(1)=c , constant time
$T(n)=2T(n/2)+\Theta(1)$

We are in the case 1. of Master Theorem with a= b=2, since the function f(n) in Master Theorem is $=O(n^{\log_2 2-\varepsilon})= O(1)$, with $\varepsilon=1$. Thus $T(n)= \Theta(n^{\log_2 2})= \Theta(n)$.

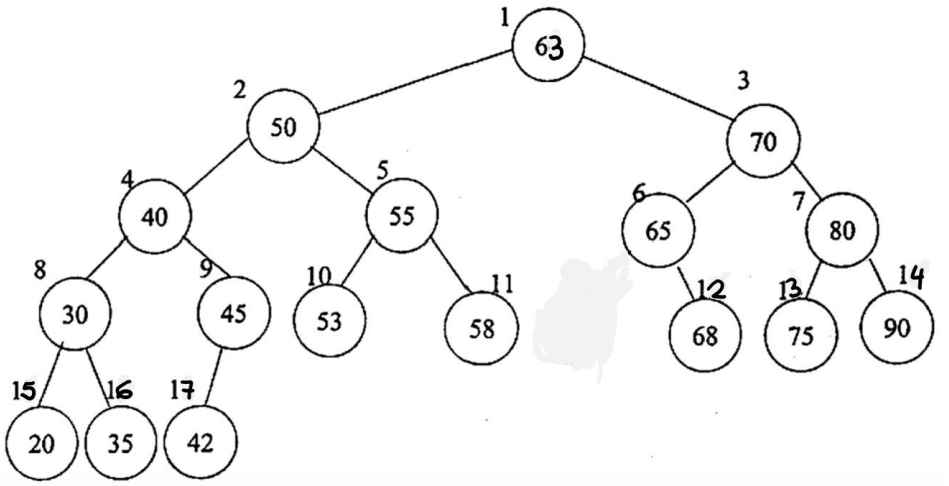
2P for correct formula for T(n) 4P for application of Master Theorem.

Exercise 5 (4P)
Consider the binary search tree below. What does it happen, if we apply the pseudocode TREE DELETE to the node z with key 60, i.e. the root of the tree.




```
TREE-DELETE(T, z)
1  if z.left == NIL
2      TRANSPLANT(T, z, z.right)
3  elseif z.right == NIL
4      TRANSPLANT(T, z, z.left)
5  else y = TREE-MINIMUM(z.right)
6      if y.p ≠ z
7          TRANSPLANT(T, y, y.right)
8          y.right = z.right
9          y.right.p = y
10     TRANSPLANT(T, z, y)
11     y.left = z.left
12     y.left.p = y
```

Answer:



Exercise 6 (6P)

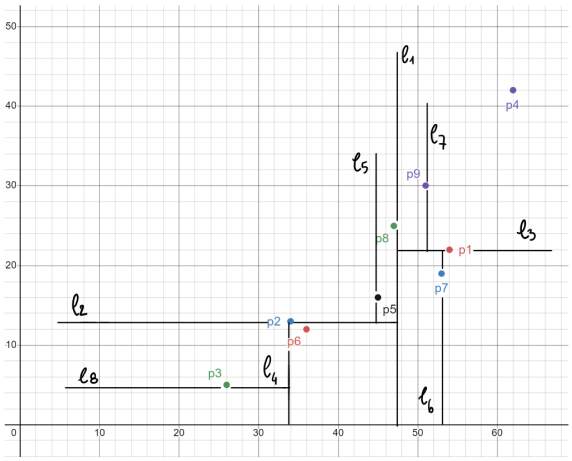
Consider a data set of 9 people P_i being x_i years old and y_i years of service in their company.

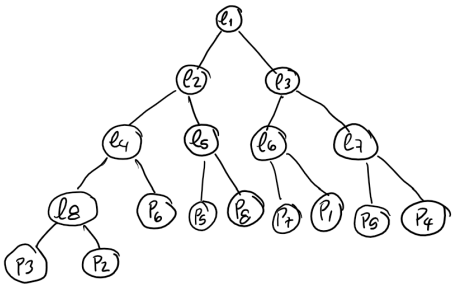
i	1	2	3	4	5	6	7	8	9
x_i	54	34	26	62	45	36	53	47	51
y_i	22	13	5	42	16	12	19	25	30

By using the pseudocode seen in the lectures, build a KD Search Tree with leaves the 9 people $P_1,\ldots,P_9$.

Algorithm BUILDKDTREE(P, *depth*)
Input. A set of points P and the current depth *depth*.
Output. The root of a kd-tree storing P.
1. **if** P contains only one point
2. **then return** a leaf storing this point
3. **else if** *depth* is even
4. **then** Split P into two subsets with a vertical line ℓ through the median x-coordinate of the points in P. Let P_1 be the set of points to the left of ℓ or on ℓ, and let P_2 be the set of points to the right of ℓ.
5. **else** Split P into two subsets with a horizontal line ℓ through the median y-coordinate of the points in P. Let P_1 be the set of points below ℓ or on ℓ, and let P_2 be the set of points above ℓ.
6. $v_{\text{left}} \leftarrow$ BUILDKDTREE(P_1, *depth* + 1)
7. $v_{\text{right}} \leftarrow$ BUILDKDTREE(P_2, *depth* + 1)
8. Create a node v storing ℓ, make v_{left} the left child of v, and make v_{right} the right child of v.
9. **return** v

Answer:

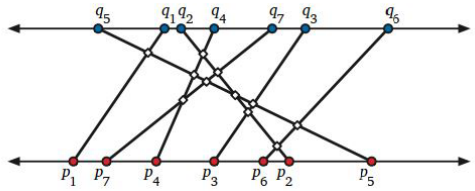




3P correct line divisions 3P correct tree.

Exercise 7 (10P)

Consider in a Cartesian plan {(x,y) | x and y are real numbers} two sets of numbers, $\{p_1, p_2,\ldots,p_n\}$ on the line with equation y=0 and $\{q_1, q_2,\ldots,q_n\}$ on the line with equation y=1. Let s_i be the segment joining the point p_i with q_i. See the picture below



We may assume that the two series of points consist of n distinct points each. Describe an algorithm, which counts the number of pairs of segments s_i, which intersect each other. Analyse the running time of your algorithm. You will receive the whole points assigned in this exercise if your algorithm is correct and of class O(n log n). In case it is correct and of $O(n^2)$ you will receive half of the points.

A possible answer: We denote by **i** the segment joining the point $\mathbf{p_i}$ with the point $\mathbf{q_i}$. the we sort the array $\{p_1, p_2,\ldots,p_n\}$ according the x coordinate, where we start on the left with the point with the smallest x coordinate. Now permute accordingly the above sorting process the points in $\{q_1, q_2,\ldots,q_n\}$ in order to do not change the above describe segments **i**. Thus we can now assume that the point $\{p_1, p_2,\ldots,p_n\}$ have coordinates such that $x_1<x_2<\ldots<x_n$ and the segments **i** joining $\mathbf{p_i}$ with $\mathbf{q_i}$ did not change. Now let us denote by b_i the x coordinate of q_i. With this new ordering of the points, for any indices i<j, the segments i and j intersect if and only if $b_i>b_j$. We call a pair (q_i, q_j) satisfying i<j and $b_i>b_j$ an inversion. Therefore, we have to calculate the total number of inversions. The algorithm that we present will count the number of inversions and at the same time sort the set $\{q_1, q_2,\ldots,q_n\}$according the values of their x coordinates.

Now recursively count all inversions in $\{q_1, q_2,\ldots,q_m\}$and sort them, where $m=\lfloor n/2 \rfloor$, and recursively count all inversions in $\{q_{m+1}, q_2,\ldots,q_n\}$and sort them, then make a sort of merge of the two above results: for i=1 consider all indexes j from m+1 to n such that $b_j<b_1$, since $\{q_{m+1}, q_2,\ldots,q_n\}$ is sorted (according their b value, which is the x coordinate) the set of the index veryfing $b_j<b_1$ is either an empty set or is a consecutive set of indexes m+1, …, j_1. In the new array you put $q_{m+1},\ldots,q_{j1}$ before q_1 and count j_1-m inversions (the number of indexes between j_1 and m+1). Then reiterate the procedure with q_2 and the index from j_1 the possible q_k having x coordinate $b_j<b_2$ after q_1 and before q_2 and add this number of element in the counting of the inversions and so on. Since $\{q_1, q_2,\ldots,q_m\}$ and $\{q_{m+1}, q_2,\ldots,q_n\}$ are two sorted set according the x coordinate, this merge operation cost O(n) running time. Therefore the recursive formula for the running time T(n) is

$$T(n)=2T(n/2)+O(n)$$

Master Theorem tell us that T(n) is in class O(nlog n).

5P for a partial correct alg, 10P if complete correct, just mention to ANYSEGMENTINTERSEC 3P