MSE_FTP_Alg.HS2023
Final-PART1 (JK Canci)
Tuesday, 06.02.2024, 9.15-11.15

Name, Last name: ______________________

E-mail: ______________________

UAS: ______________________

Part of UAS: ______________________

General information

- 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.
- Communication with others during the exam is forbidden. Mobile phones must be turned off. No electronic tools are allowed.
- Do not write with red or green pens.
- Open Book: that means books and copy of slides are allowed (solutions to exercises excluded).
- Justify your answers.

Exercise	max. points	reached points
Exercise 1	4	
Exercise 2	5	
Exercise 3	7	
Aufgabe 4	6	
Exercise 5	9	
Exercise 6	9	
Total:	40	

Exercise 1 — 4 Points

In the following pseudocode we give as an input a positive integer $n > 2$. Describe in words what does the algorithm. What is the output if $n = 5$. Calculate the Θ-class of the generic running time $T(n)$.

Algorithm 1 Algorithm

INPUT: a positive integer $n > 2$ $(c_1 n)$
Create A an empty array with n entries
Set $A[1] = A[2] = 1$ (c_2)
for $(i = 3;\ i \leq n)$ **do** ($n - 3$ times)
    $A[i] = A[i-1] + A[i-2]$ (c_3)
return $A[n]$ (c_4)

Solution: The algorithm build the so called Fibonacci sequence

$$1, 1, 2, 3, 5, 8, 13, \ldots$$

where starting from the third number, in any position we have the sum of the previous two.

Therefore we $n = 5$ the output is $(1, 1, 2, 3, 5)$

in the following c_i denotes a constant for each index i. In red the costs.

correct output 2P

correct running time 2P

Exercise 2 — 5 Points

In the following pseudocode we give as an input a list of n points with $n \geq 2$

$$(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)$$

Describe in words what does the algorithm. Calculate the Θ-class of the generic running time $T(n)$.

Solution: The algorithm returns the smallest distance between two points (3P) the running time is $\Theta(n^2)$ (2P).

Exercise 3 — 7 Points

For a given increasing ordered array A (of numbers) write an algorithm to check if a number k is an entry (element) of the ordered array A (it returns the index of the element if present, or -1 if not present). Write a recursive algorithm (pseudocode) with the following steps (in divide-and-conquer style):

- **Decomposition**: divide the array in half and identify the portion in which will be contained the element to be searched.

Algorithm 2 Algorithm(n-points)

INPUT: $n \geq 2$ points $(x_1, y_1), (x_2, y_2) \dots, (x_n, y_n)$

$$a = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

for $(i = 1;\ i \leq n)$ **do**
  for $(j = i + 1;\ j \leq n)$ **do**
    $r = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
    if $(r < a)$ **then**
      $a = r$
return a

- **Recursion**: recursively search for the element in one of the two parts.
- **Recombination**: return the result of the recursive call as is

Write the pseudocode and calculate the running time of your algorithm. The faster is your algorithm and the higher is the mark you will receive.

Solution: Correct idea 3 P, correct code 3 P fastest possible 1P

Aufgabe 4 — 6 Points

(4P) Write in increasing order from left to right the following list of Theta classes. When two classes are equals, make it clear.

$$\Theta(n\sqrt[3]{n}), \Theta(n^{4/5}), \Theta(n^{3/2}), \Theta(n\sqrt[2]{n}), \Theta(n\log_4 n), \Theta(e^{(\sqrt{2})\cdot n}), \Theta(e^n), \Theta(100n), \Theta(n!)$$

(2P) Give two functions $f(n)$ and $g(n)$ such that

$$\Theta(f(n) + g(n) \neq \Theta(f(n)) \text{ and } \Theta(f(n) + g(n)) \neq \Theta(g(n))$$

Solution: (0.5 P each correct relationship)

$$\Theta(n^{4/5}), \Theta(100n), \Theta(n\log_4 n), \Theta(n\sqrt[3]{n}), \Theta(n^{3/2}) = \Theta(n\sqrt[2]{n}), \Theta(e^n), \Theta(e^{(\sqrt{2})\cdot n}), \Theta(n!)$$

$f(n) = n$ and $g(n) = -n + 1$

Exercise 5 — 9 Points

Consider the weighted oriented graph given by the following adjacency matrix

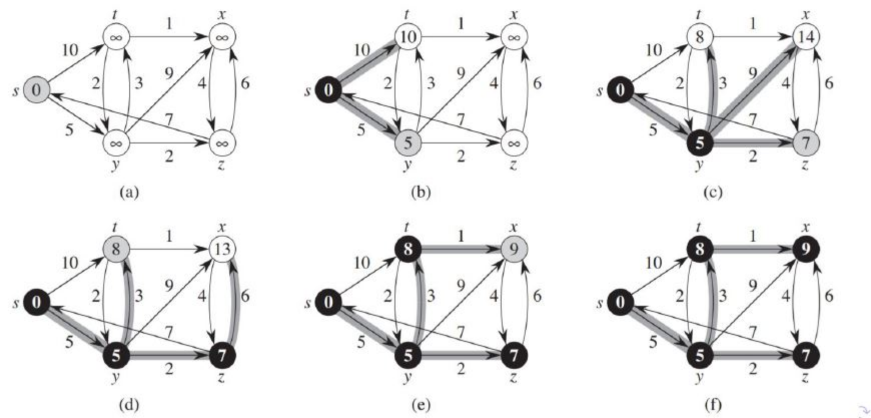
$$\begin{array}{c c} & \begin{array}{ccccc} s & x & y & z & w \end{array} \\ \begin{array}{c} s \\ x \\ y \\ z \\ w \end{array} & \begin{pmatrix} 0 & 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix} \end{array}$$

(2P) Draw the graph.

(7P) If possible apply Dijkstra's algorithm. Described step by step how the algorithm works on the given graph, in other words describe it as done in the figure below

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

The shortest–path estimates appears within the vertices. Shaded edges indicates predecessor values. Black vertices are in S. White vertices are in the min-priority queue $Q = V \setminus S$. The vertex in grey is the EXTRACT–MIN(Q).



> **Solution:** correct graph 2
>
> second part:
>
> only describe with words but no answer or almost completely worng 2P
>
> If the solution is not correct but almost 4P

Exercise 6 — 9 Points

Consider the following points in the xy-plane:

$$(-5,5), (-4,3), (-3,-5), (-2,-1), (-1,0), (0,-3), (1,2), (2,-4), (3,6)$$

By using the following pseudocode, draw a KD Tree with input the above 9 Points and depth 0.

Algorithm BUILDKDTREE(P, $depth$)
Input. A set of points P and the current depth $depth$.
Output. The root of a kd-tree storing P.

if P contains only one point
  then return a leaf storing this point
  else if $depth$ is even
    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.
    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.
  $v_{\mathrm{left}} \leftarrow$ BUILDKDTREE(P_1, $depth + 1$)
  $v_{\mathrm{right}} \leftarrow$ BUILDKDTREE(P_2, $depth + 1$)
  Create a node v storing ℓ, make v_{left} the left child of v, and make v_{right} the right child of v.
  return v

Solution: Correct points in the plane 1P
correct subdivison 5P (only partially 2P)
correct graph 3P