FTP_Algorithms - Final Exam
Part 1 (J.K. Canci)
February the 8th 2022, 14.15-16.15

Family name: _____

Given name: _____

E-mail: _____

General information

- Duration 120 minutes in total for both parts.
- The task of the exam will be to fill the bulletin by means of a normal pen. For some of the exam questions you may want to add computations, figures, etc. on additional sheets of paper.
- Immediately after the exam, the filled paper bulletin and all the additional solution sheets must be submitted to the teachers.
- Put in front of you on the table your student's card and your COVID certificate.

Permissible aids during the exam

- Open book, open computer (you basically have access to any resource).
- We will be very strict with any attempts of communication during the exam. So please make sure that during the entire exam, at NO time apps or windows which could enable to communicate with other people are visible on your screen.
- You are not allowed to use electronic pens and apps like OneNote etc. since it was too easy to exchange information that way.
- Please make sure that you have the necessary technical equipment ready (laptop or tablet computer, charging cable, books, documents, plain paper, pens, etc).

Languages allowed for your answers:

- EN, DE

Exercise	max. points	achieved points
Exercise 1	15	
Exercise 2	15	
Exercise 3	15	
Exercise 4	15	
Total:	60	

Exercise 1 — 15 Points

Running time

1. (6P) Write the following Theta-classes in non decreasing order from left to right (i.e. smallest on the left):

   $$\Theta(n^2), \Theta(e^{\log_2 n}), \Theta(\sin n), \Theta(\ln^2 n), \Theta(\log_2 n), \Theta(n^3 + \ln(n^2)), \Theta(e^{(n^2)}), \Theta(e^{\ln(n^4)})$$

   As usual ln denotes the natural logarithm (i.e. in base e).
2. (3P) We consider a recursive algorithm that involves an input with n objects and having running time $T(n) = 4 \cdot T(n/2) + n^2$. Determine the running Theta class of the algorithm.
3. (3P) We consider a recursive algorithm that involves an input with n objects and having running time $T(n) = 4 \cdot T(n/2) + n$. Determine the running Theta class of the algorithm.
4. (3P) We consider a recursive algorithm that involves an input with n objects and having running time $T(n) = 4 \cdot T(n/2) + n^3$. Determine the running Theta class of the algorithm.

Justify all your answers.

Solution:

1. $$\Theta(\sin n), \Theta(\log_2 n), \Theta(\ln^2 n), \Theta(n^2), \Theta(e^{\log_2 n}), \Theta(n^3 + \ln(n^2)), \Theta(e^{\ln(n^4)} \Theta(e^{(n^2)}))$$

   (each wrong inequality -1P)
2. Case 2 Master theorem. $\Theta(n^2 \ln n)$.
3. Case 1 Master theorem. $\Theta(n^2)$.
4. Case 3 Master theorem. $\Theta(n^3)$. (In all three cases. 1p corrected answer 2p correct justification)

Exercise 2 — 15 Points

Pseudocode. In the following requested pseudocode you may use/call any pseudocode considered in our lectures.

1. (6P) Write a pseudocode, which takes the input $A = \{a_1, a_2, \ldots, a_n\}$ and gives you as output $B = \{a_n, \ldots, a_2, a_1\}$ (i.e. same entries as in A but, in the reversed order).
2. (9P) Let S be a set with n integers, randomly ordered and not necessarily distinct. Let m be an integer. Write a pseudocode, which tests whether there are two elements $a, b \in S$ with $a + b = m$ (a and b may be the same number). Determine the running time of your algorithm.

Solution:

1. One possible answers is the following one.

   The input of the following code is an Array A with n entries

---

REVERSEARRAY(A)
for $i = 1$ to $\lfloor n/2 \rfloor$
  exchange $A[i]$ with $A[n-i+1]$
return A

The pseudocode swaps the the first entry with the last one, the second with the second last and so on. The running time is $\Theta(n)$, because we have about n/2times the same operation, which costs constant time, let say c.

(6 points correct, 3P if something is correct but not completely)

2. One possible answers is the following one.

   The input of the following code is an Array S with n integers as entries and an integer m

---

SUMTEST(S,m)
for $i = 1$ to $n-1$
  for $j = i+1$ to n
    if $S[i] + S[j] = m$ return true

With $i = 1$ it makes $n-1$ sums, with $i = 2$ it makes $n-2$ sums and so on. So we have

$$(n-1) + (n-2) + \ldots + 1 = \frac{n(n-1)}{2}$$

So the above pesudocode is $\Theta(n^2)$. The above psudocode is intuitive but, we can do better.

SUMTESTBEST(S,m)
$i = 1$ to $j = n$
  if $S[i] + S[j] = m$ return true
  if $S[i] + S[j] < m$, i=i+1
  else $j = j+1$ (# that is if $S[i] + S[j] > m$)
stop the test when it returns true or $i > j$

The running time is $\Theta(n)$.

(6 point ofr the code, 3P if something is correct but not completely, and 3 for the running time)

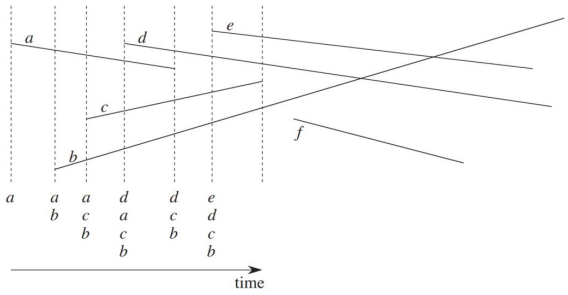
Exercise 3 — 15 Points

Computational geometry.

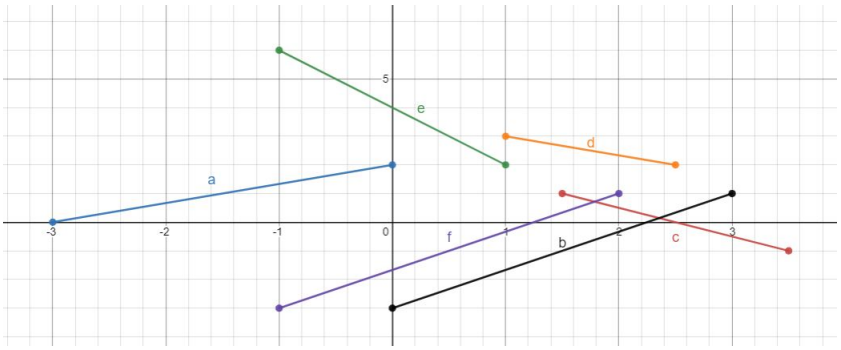
1. We have seen the algorithm ANYSEGMENTINTERSECT:

```
ANY-SEGMENTS-INTERSECT(S)
 1  T = ∅
 2  sort the endpoints of the segments in S from left to right,
        breaking ties by putting left endpoints before right endpoints
        and breaking further ties by putting points with lower
        y-coordinates first
 3  for each point p in the sorted list of endpoints
 4      if p is the left endpoint of a segment s
 5          INSERT(T, s)
 6          if (ABOVE(T, s) exists and intersects s)
                or (BELOW(T, s) exists and intersects s)
 7              return TRUE
 8      if p is the right endpoint of a segment s
 9          if both ABOVE(T, s) and BELOW(T, s) exist
                and ABOVE(T, s) intersects BELOW(T, s)
10              return TRUE
11          DELETE(T, s)
12  return FALSE
```

which is based on sweep lines as pictured in the following



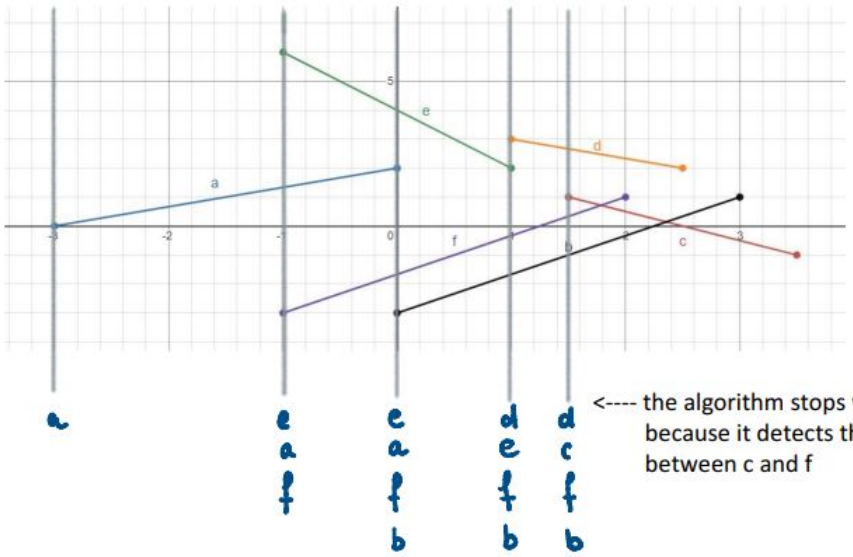
We consider the following six segments a, b, c, d, e and f on the (x, y)-plane represented in the picture below:



Draw in the above picture the sweep lines, that the pseudocode needs to check in order to determine whether there are any two segments which intersect. Do not forget to write the partial order of segments at each requested sweep line.

Solution:

1.



a

e a f

e a f b

d e f b

d c f b

<---- the algorithm stops with this sweep line because it detects the intersection between c and f

In the text of the exam was requested to list the partial order at each sweep line, this order was defined at slide 8 of "Lecture 11 and 12". The algorithm "AnyTwoSegmentsIntersect" consider a way to add and eliminate elements in the partial order, that do not always corresponds to the complete partial order.

5 poins are for giving the exact set of the sweep lines. 2 points are for each exact partial order.

(5 points for last sweep line, 2P for each corrected order at sweep lines)

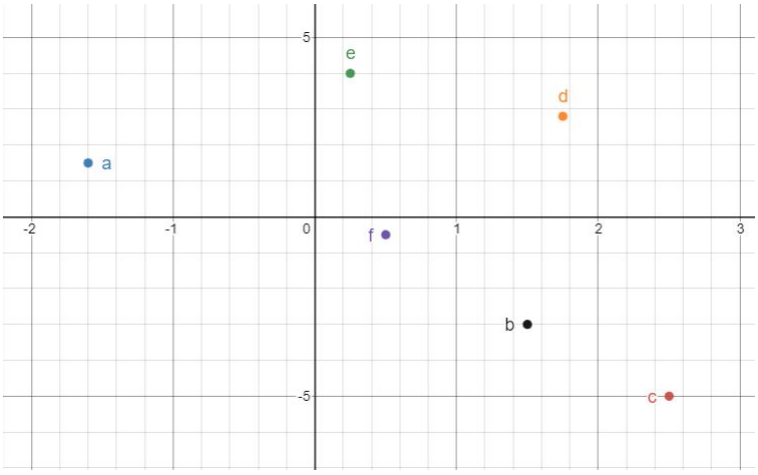
Exercise 4 _____ 15 Points

Points on the plane.

1. (9P) Write a pseudocode, which takes as input a set S containing n points of the (two dimensional) (x, y)-plane and verifies whether three points of the inputs are collinear (i.e. all three contained in a same straight line). Your algorithm has to be $O(n^2 \ln n)$. [Hint: call a pseudocode considerd in the lectures].
2. (6P) By using the algorithm BUILDKDTREE

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

on the following set of six points a, b, c, d, e and f



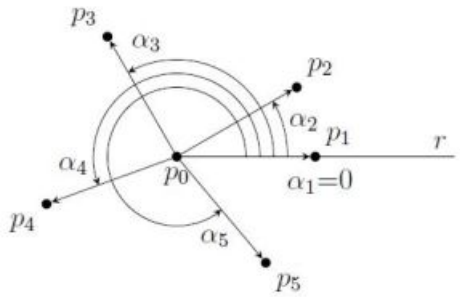
draw the corresponding KD-Tree.

Solution:

1. The idea was to use a pseudocode given in the solutions of exercise week 6 (of HS2021). I think this is the unique a little difficult exercise. Solving all the other exercises correct plus just some idea of this would give for sure an A.

   Here we present another solution, which is self contained.

   Just to fix ideas consider the set of points pictured below



   We choose a point among the n ones. In the picture take p_0. Now consider the angles forming from the segment $\overline{p_0p_1}$ and each other segment $\overline{p_0p_j}$ starting from p_0. To calculate the angles we use DIRECTION(p_0, p_j, p_1) which returns the cross product $(p_1 - p_0) \times (p_j - p_0)$. Recall that the cross product can be calculated with the formula

$$(p_1 - p_0) \times (p_j - p_0) = ||(p_1 - p_0)|| \cdot ||(p_j - p_0)|| \cdot \sin \alpha_j$$

   Therefore by using the inverse function $\sin^{-1}$ one can calculate the angles α_j. All these calculations have as costs a constant c.

   For each center p_0 (after having chosen a point p_1) one has to iterate $n-2$ times the calculation with the cross product.

Now order the angles α_i in increasing order, this cost $O(n \log n)$ (use for example merge-sort).

Then use the pseudoalgorithm SUMTESTBEST (of Ex.2) to check whether the difference of two angles is zero or π, in both case, if this happens with α_i and α_j, then P_0, P_i, P_j are collinear .SUMTESTBEST has running time $O(n)$ Also the procedure with center in one of the points of the set (in our first case with P_0) has running time

$$c \cdot (n-2) + O(n \log n) + O(n) = O(n \log n)$$

One has to repeat the procedure (in the worst case) for all points $P_0, P_1, \ldots, P_{n-1}$. Also the running time is $n \cdot O(n \log n) = O(n^2 \log n)$.

(9 points)

2. (2 points corrected splitting lines. 4 points correct tree)