ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Midterm Exam, Algorithms 2017-2018

- You are only allowed to have a handwritten A4 page written on both sides.
- Communication, calculators, cell phones, computers, etc... are not allowed.
- Your explanations should be clear enough and in sufficient detail that a fellow student can understand them. In particular, do not only give pseudo-code without explanations. A good guideline is that a description of an algorithm should be such that a fellow student can easily implement the algorithm following the description.
- You are allowed to refer to algorithms covered in class without reproving their properties.
- **Do not touch until the start of the exam.**

Good luck!

Name: ______________________ **N° Sciper:** ______________

Problem 1 / 27 points	Problem 2 / 18 points	Problem 3 / 28 points	Problem 4 / 27 points

Total / 100

1 *(27 pts)* **Basic questions.** This problem consists of three subproblems.

1a *(8 pts)* Give tight asymptotic bounds for the following recurrences (assuming that $T(1) = \Theta(1)$):

(i) $T(n) = 2T(n/4) + \Theta(\sqrt{n})$

(iii) $T(n) = 2T(n-2) + \Theta(1)$

(ii) $T(n) = 4T(n^{1/8}) + \Theta(\log n)$

(iv) $T(n) = 16T(n/4) + \Theta(n^2)$

1b *(9 pts)* Answer true/false questions below (each question worth 1 point):

A binary tree of height $h \geq 1$ has at most 2^h nodes (recall that a tree with a single node has height 0). True or False?

The worst-case complexity for searching in a binary search tree is $O(\log n)$. True or False?

A max-heap can be built from an unsorted array $A[1...n]$ in time $O(n)$. True or False?

Extracting the maximum element from a max-heap has worst-case runtime $\Omega(n)$. True or False?

If $f(n) = n^{2.1}$ and $g(n) = n^2 \log n$, then $f(n) = \omega(g(n))$. True or False?

If $f(n) = 2^{\sqrt{\log n}}$ and $g(n) = \log^2 n$, then $f(n) = o(g(n))$. True or False?

An array of size n which contains only zeros and ones can be sorted in linear time using a constant amount of additional memory. True or False?

If every node in a binary tree has either 0 or 2 children, then the tree has height $O(\log n)$. True or False?

Running merge sort on a sorted array takes $O(n)$ time. True or False?

1c *(10 pts)* In this problem you are given the code of a function UNKNOWN(str) that takes as input a string and outputs **true** or **false**.

```
UNKNOWN(str)

Initialize an empty stack S
n=str.length
for i=1 to n
    if str[i]=='A' or str[i]=='C'
        PUSH(S, str[i])
    else if str[i]=='B'
        if STACK-EMPTY(S) or POP(S)!='A'
            return false
    else if str[i]=='D'
        if STACK-EMPTY(S) or POP(S)!='C'
            return false
if STACK-EMPTY(S)
    return true
else
    return false
```

What does UNKNOWN output on inputs below?

1. UNKNOWN("ABBA")=
2. UNKNOWN("ACBD")=
3. UNKNOWN("ABCD")=
4. UNKNOWN("AAAABBBBAAAA")=
5. UNKNOWN("ACDBABCDCDCCDD")=

2 *(18 pts)* **Recurrences.** Consider the following algorithm UNKNOWN that takes as input an integer n:

```
UNKNOWN(n):

if n < 10
    return
UNKNOWN(⌊4n/5⌋)
for i = 1 to n
    for j = 1 to i
        print "Almost done!"
UNKNOWN(⌊3n/5⌋)
return
```

2a *(4 pts)* Let $T(n)$ be the time it takes to execute UNKNOWN(n). **Give the recurrence relation** for $T(n)$. To simplify notation, you may assume that $n/5$ always evaluates to an integer.

2b *(14 pts)* **Prove** tight asymptotic bounds on $T(n)$. Specifically, show that $T(n) = \Theta(n^a \log n)$ for some integer $a \geq 0$. You may simplify your calculations by assuming that $\lfloor n/5 \rfloor = n/5$.

3 *(28 pts)* **Crater crossing.** As you may (or may not) have heard in the news, the famous Fiery Crater in the beautiful Swiss Alps has just been opened to the public, and naturally a number of companies are now trying to establish Tyrolean routes across the crater. Each of the n companies designated a pair of climbers to set up the route: for every $i = 1, \ldots, n$, the i-th pair of climbers occupies distinct positions s_i, t_i along the rim of the crater. The rim of the crater is a perfect circle, and $s_i, t_i \in [0, 2\pi)$ correspond to the angle that the climbers in the i-th pair are positioned at (see Fig. 1). Every pair of climbers is connected by a tight rope (basically a straight line), which is the candidate route. There is a major problem, however: the routes intersect! Since nobody wants to be part of a mid-air collision above a sea of lava, the Swiss Alpine Guides decided to open a subset of routes that do not intersect, and are hence considered safe. Each route also has a non-negative fun parameter f_i, $i = 1, \ldots, n$, and the Mountain Guides would like to open a non-intersecting subset of routes that maximizes the total fun. They need your help.

Input: A collection of n pairs $s_i, t_i \in [0, 2\pi)$ specifying positions of pairs of climbers on the rim of the crater, for $i = 1, \ldots, n$. The fun parameters f_i, $i = 1, \ldots, n$, for each of the n routes. You can assume that no two climbers occupy the same position on the rim of the crater.

Output: The maximum possible total fun (sum of fun parameters) achievable by a non-intersecting subset of routes.

An example problem instance is given in Fig. 1 below. In this instance $n = 4$, and the fun parameters of the 4 routes are $f_1 = f_2 = 7$, $f_3 = 1$ and $f_4 = 10$. The optimal solution opens routes 1, 2 and 3, and the total fun is $7 + 7 + 1 = 15$.

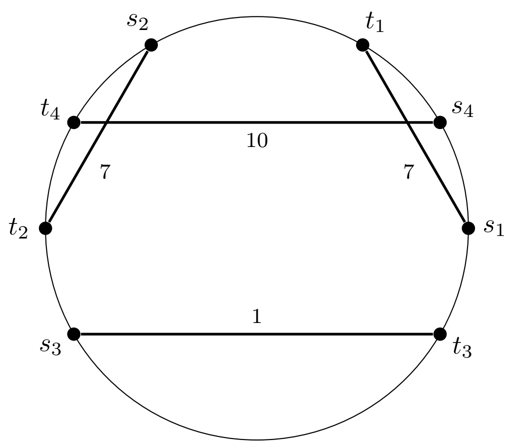


Figure 1. Illustration of set of candidate routes $s_i, t_i, i = 1, \ldots, 4$, where $s_1 = 0$, $t_1 = \pi/3$, $s_2 = 2\pi/3$, $t_2 = \pi$, $s_3 = 7\pi/6$, $t_3 = 11\pi/6$, $s_4 = \pi/6$, $t_4 = 5\pi/6$. The fun parameters are $f_1 = f_2 = 7$, $f_3 = 1$, $f_4 = 10$. The optimal solution opens routes 1, 2 and 3, and the total fun is $7 + 7 + 1 = 15$.

In the following we will design and analyze an efficient algorithm that finds the largest total fun achievable by a non-intersecting subset of routes.

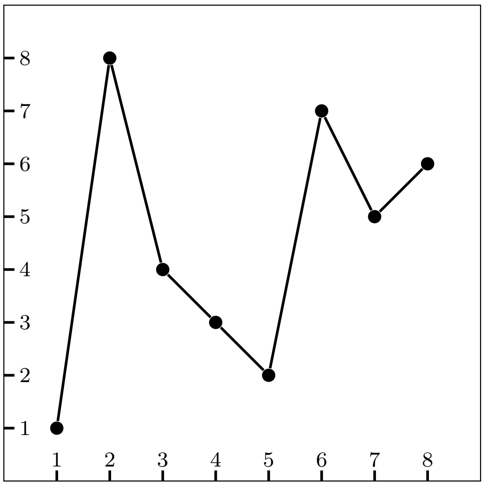
Let $p_1, \ldots, p_{2n}$ denote the $2n$ distinct positions that the climbers occupy along the rim of the crater, in counterclockwise order starting from an arbitrary climber. In the example in Fig. 1, if we start with s_1 and traverse the positions of the climbers in counterclockwise order, we get $p_1 = 0$, $p_2 = \pi/6$, $p_3 = \pi/3$, $p_4 = 2\pi/3$, $p_5 = 5\pi/6$, $p_6 = \pi$, $p_7 = 7\pi/6$, $p_8 = 11\pi/6$. For every $1 \leq i \leq j \leq 2n$ let $c[i, j]$ denote the maximum total amount of fun that can be achieved by opening a non-intersecting set of routes whose endpoints belong to the set $\{p_i, p_{i+1}, \ldots, p_j\}$. Note that $c[1, 2n]$ is the solution that you are asked to find.

3a *(23 pts)* Explain how to express $c[i, j]$ recursively in terms of values $c[a, b]$ for $i < a \leq b \leq j$. Write down the recurrence relation together with the base case. You may assume that you have access to a function Pair(i) that, given an index $i \in \{1, 2, \ldots, 2n\}$, in $O(1)$ time outputs the index in p of the position of the climber that the climber in position p_i is paired to. In the example above Pair(2)=5 and Pair(5)=2, since the climber in position 2 is paired to the climber in position 5 (since $p_2 = \pi/6 = s_4$ and $p_5 = 5\pi/6 = t_4$).

3b *(5 pts)* What is the runtime of the bottom-up implementation of the dynamic programming solution to the problem that uses your recurrence from **3a**? Justify your answer.

4 *(27 pts)* **Tallest mountains.** You are planning a hike in the beautiful Swiss Alps again, and are facing a difficult choice: which mountain range should you go to to maximize opportunities for fun hikes? In this problem you will design an algorithm for this challenging task.

A mountain range with n mountains can be represented by an array of mountain heights A of length n, where $A[i]$ for $i = 1, \ldots, n$ is the height of the i-th mountain on the horizon from left to right – see Fig. 2 for an illustration.



Array $A[1 \ldots 8] =$

1	8	4	3	2	7	5	6

Figure 2. Representation of a mountain range as an array A of mountain heights.

A mountain range with n mountains offers $n(n+1)/2$ different hikes: for every $1 \leq i \leq j \leq n$ you can start at the i-th mountain and then visit all mountains $i, i+1, \ldots, j$ in a single trip. The height h_{ij} of a hike from mountain i to mountain $j \geq i$ is the height of the tallest mountain that you visit along the way, i.e. for $1 \leq i \leq j \leq n$ we define

$$h_{ij} := \max_{i \leq k \leq j} A[k].$$

The total height of a mountain range is sum of heights of all hikes $1 \leq i \leq j \leq n$, i.e. $\sum_{i=1}^{n} \sum_{j=i}^{n} h_{ij}$. Your task is to **design** and **analyze** an efficient algorithm for computing the total height of a mountain range.

> **Input:** An array A of integers of length n. You can assume that all elements of A are distinct.
>
> **Output:** The total height of A: $\sum_{i=1}^{n} \sum_{j=i}^{n} h_{ij}$, where $h_{ij} = \max_{i \leq k \leq j} A[k]$.

A solution that runs in $O(n \log n)$ time suffices for full credit (e.g. there exists a divide and conquer approach similar to what is used for the maximum subarray problem). $O(n)$ time solutions also exist.

4a *(22 pts)* Design an efficient algorithm for computing the total height of an array A of n integers.

4b *(5 pts)* Give a tight asymptotic bound on the runtime of your algorithm.