ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Midterm Exam, Algorithms 2016-2017

- 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 / 20 points	Problem 2 / 30 points	Problem 3 / 28 points	Problem 4 / 22 points

Total / 100

1 *(20 pts)* **Asymptotic growth and heaps.**

1a *(10 pts)* Arrange the following functions in increasing order of asymptotic growth:

$$2^n, \quad n^2 + n/3, \quad n \log n, \quad \log_2 \log_2 n, \quad n^{100/\log_2 n}, \quad n!, \quad 4^{\sqrt{n}}, \quad \sqrt{3^n + 2^n}$$

1b *(10 pts)* **Draw** the heap that results from executing BUILD-MAX-HEAP(A, n) on input

$A =$

3	1	6	7	2	10	5	14	12

and $n = 9$.

2 *(30 pts)* **Recurrences.** Consider the following algorithm UNKNOWN that takes as input an integer n:

```
UNKNOWN(n):

if n < 50
      return
q = ⌊n/3⌋
UNKNOWN(q)
UNKNOWN(n − q)
for i = 1 to q
      print "I love recurrences!"
UNKNOWN(n − q)
```

2a *(10 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 $n/3$ always evaluates to an integer.

2b *(20 pts)* **Prove** tight asymptotic bounds on $T(n)$. Specifically, show that $T(n) = \Theta(n^a)$ for some constant a. You may simplify your calculations by assuming that $\lfloor n/3 \rfloor = n/3$.

3 *(28 pts)* In this problem you will design an efficient algorithm for measuring distance between two strings. Your input is two strings $S = (s_1, s_2, \ldots, s_n)$ and $T = (t_1, t_2, \ldots, t_m)$, where n and m are the lengths of S and T respectively. Our measure of distance between S and T is the smallest number of deletions, insertions or substitutions that one has to make to T to make it equal to S.

For example, the distance between the strings $S =$ 'albatros' and $T =$ 'abbbas' is 5, via the sequence of operations shown below:

$$abbbas \xrightarrow{\text{delete 'b'}} abbas \xrightarrow{\text{substitute 'b' with 'l'}} albas \xrightarrow{\text{insert 't'}} albats \xrightarrow{\text{insert 'r'}} albatrs \xrightarrow{\text{insert 'o'}} albatros$$

3a *(8 pts)* Suppose that only substitutions, but no insertions and deletions, are allowed, and the strings S and T have the same length n. Give an algorithm for computing the minimum number of substitutions that are needed to turn T into S. For full credit your algorithm should run in $O(n)$ time.

3b *(20 pts)* Now suppose that all three operations (insertion, deletion, substitution) are allowed, and the strings $S = (s_1, s_2, \ldots, s_n)$ and $T = (t_1, t_2 \ldots, t_m)$ are of length n and m respectively, with n possibly different from m.

For $i = 0, 1, \ldots, n$ and $j = 0, 1, \ldots, m$ let $S_i = (s_1, s_2, \ldots, s_i)$ denote the prefix of S of length i, and $T_j = (t_1, t_2, \ldots, t_j)$ denote the prefix of T of length j.

Let $c[i, j]$ denote the distance between S_i and T_j. The initial conditions are

- $c[i, 0] = i$ for all $i = 0, 1, \ldots, n$;
- $c[0, j] = j$ for all $j = 0, 1, \ldots, m$.

The distance between S and T is $c[n, m]$.

Find a recurrence relation for $c[i, j]$. In addition, give a tight runtime analysis of the standard bottom-up implementation for finding the distance using your recurrence.

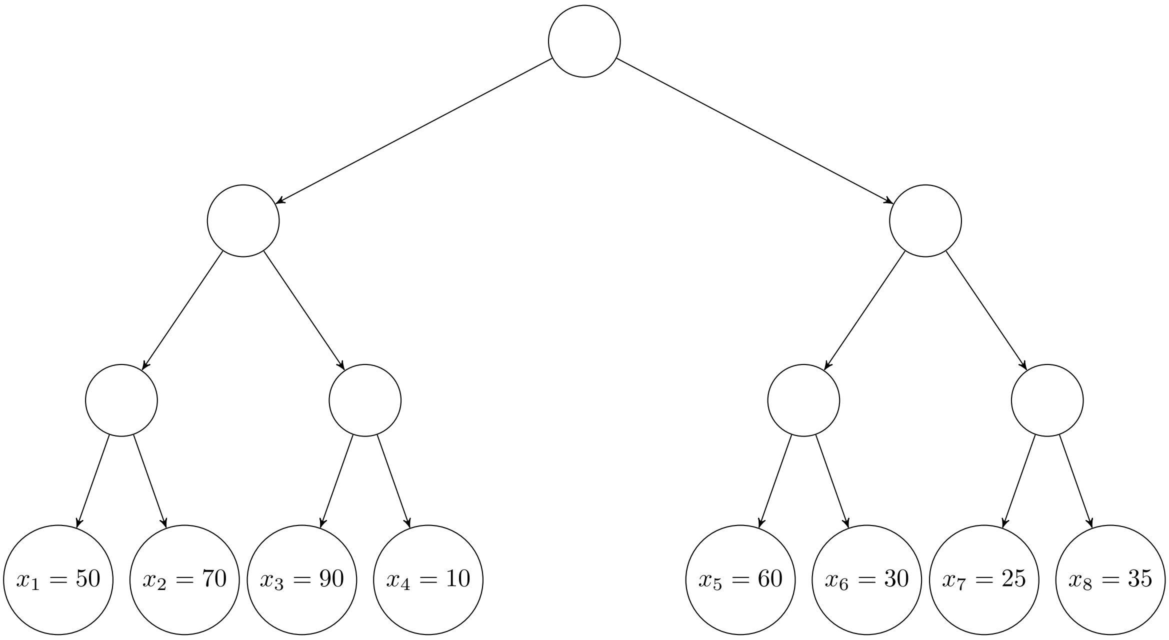

Figure 1. Illustration of the binary tree with $n = 8$ leaves annotated with numbers $x_i, i = 1, \ldots, n$.

4 *(22 pts)* **Fast interval queries.** Suppose that you are given a **complete binary tree** with numbers $1, 2, 3, \ldots, n$ at the leaves (from left to right). In particular, n is a power of 2. You are also given numbers $x_i, i = 1, \ldots, n$ associated with the leaves (see Fig. 1). Design a data structure that stores extra information at every node of the tree and allows answering *interval queries*: for an input pair $a \leq b$ of integers between 1 and n, your data structure should be able to compute $\max_{a \leq i \leq b} x_i$.

For full credit your solution should take $O(n)$ time to prepare the data structure (the binary tree with extra information stored at the nodes), and every query should be answered in $O(\log n)$ time in the worst case.

(additional space for your solutions)

(additional space for your solutions)