ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Midterm Exam, Algorithms 2018-2019

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

Total / 100

1 *(22 pts)* **Basic questions.**

1a *(4 pts)* Answer whether the following statements are **true** or **false**.

HeapSort requires $\Omega(n)$ extra space to sort an array of length n. True or false?

If $T_H(n)$ and $T_B(n)$ denote the worst case runtime of extracting the maximum element from a max heap and a binary search tree on n nodes respectively, then $T_H(n) = \Theta(T_B(n))$. True or false?

Running HeapSort on a sorted (in the correct order) array takes $O(n)$ time. True or false?

BuildMaxHeap constructs a heap from an unsorted array of length n in time $\Theta(n)$. True or false?

1b *(10 pts)* Arrange the following functions in increasing order according to asymptotic growth.

$$n \log_2 \left(n^5 + n^2\right), \quad (\log_2 \log_2 n)^5, \quad 5^{n \log_2 n}, \quad n!, \quad n^2 \log n, \; 2^n, \; \sqrt{n^3 + n^4}$$

Solution:

1c *(8 pts)* In every subproblem below you start with the same fixed binary search tree T. You should answer **yes** or **no** for every subproblem, no explanation is needed. *You should answer* **yes** *for every claim below that is true for every starting tree T, and answer* **no** *if there exists a tree T for which the claim is false.*

- Take an element z with $z.value$ distinct from all those already in the tree. Suppose we perform TreeInsert(T, z) followed immediately by TreeDelete(T, z). Is the new tree identical to T?

- Suppose now that z is an element already contained in T. If we perform TreeDelete(T, z) followed immediately by TreeInsert(T, z) is the new tree identical to T?

- We now have two new elements z_1 and z_2 with values $z_1.value$ and $z_2.value$ distinct from each other and any already in the tree. Perform Tree-Insert(T, z_1) followed by TreeInsert(T, z_2). Do we always get the same result if we insert them in the other order?

- Suppose now we are given k new elements $z_1, z_2, \ldots, z_k$ with values distinct from each other's and from any node already in the tree. We perform the following operations:

  TreeInsert(T, z_1)
  TreeInsert(T, z_2)
  ...
  TreeInsert(T, z_k)
  TreeDelete(T, z_1)
  TreeDelete(T, z_2)
  ...
  TreeDelete(T, z_{k-1})

  Is the resulting tree the same as if we had just inserted z_k?

2 *(30 pts)* **Recurrences.** Consider the following algorithm UNKNOWN that takes as input an array A and two indices low and $high$ in the array:

```
UNKNOWN(A, low, high)
1. for i = low to high
2.       print A[i]
3. if low = high
4.       return
5. else
6.       p = low + ⌊(high − low)/√2⌋
7.       q = ⌊(low + high)/2⌋
8.       UNKNOWN(A, low, p)
9.       UNKNOWN(A, q + 1, high)
10.      UNKNOWN(A, low, q)
11.      return
```

2a *(10 pts)* Let $T(n)$ be the time it takes to execute UNKNOWN$(A, 1, n)$. **Give the recurrence relation** for $T(n)$. To simplify notation, you may ignore floors and ceilings in your recurrence.

Solution:

2b *(20 pts)* **Prove** tight asymptotic bounds on $T(n)$. Specifically, show that $T(n) = \Theta(n^a)$ for some integer $a \geq 0$.

Solution:

3 *(28 pts)* **Rod cutting revisited.** You are given a rod of length n with m weak points on it at distances $a_1,\ a_2,\ \ldots,\ a_m$ from one end. $a_1,\ a_2,\ \ldots,\ a_m$ are integers such that $0 < a_1 < a_2 < \ldots < a_m < n$. You have to cut the rod at all weak points in some order. Each time you cut a rod at some weak point it falls apart into two pieces and two smaller rods are created, so after performing m cuts you will have $m + 1$ smaller rods.

The cost of cutting a rod is equal to its length, and you want to minimize the overall cost of cutting the rod into $m+1$ pieces. In this problem your task is to design and analyze an algorithm that, given the length n of the rod and the locations of the weak points, returns the cost of the optimal solution.

> **Input:** The length n of rod (an integer), the number m of weak points, and the locations of the weak points $0 < a_1 < a_2 < \ldots < a_m < n$ (the locations $a_i, i = 1, \ldots, m$ are integers).
>
> **Output:** The smallest possible cost of cutting the rod into $m+1$ pieces at the weak points.

For example, if $n = 7$ and there are $m = 2$ weak points on the rod, with locations $a_1 = 4$ and $a_2 = 6$ respectively (see Fig. 1), the optimal solution is as follows. We first break the rod at the first weak point $a_1 = 4$, getting two rods, of length 4 and 3 respectively. We then break the rod of length 3 at the second weak point into two rods, with lengths 2 and 1 respectively. The total cost is $7 + 3 = 10$.

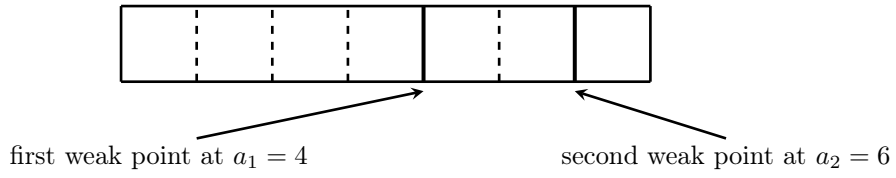

Figure 1. A rod of length $n = 7$ with $m = 2$ weak points, at locations $a_1 = 4$ and $a_2 = 6$ respectively.

Design an efficient dynamic programming algorithm for this problem and **analyze its runtime.**

Solution:

(Solution to problem 3 continued)

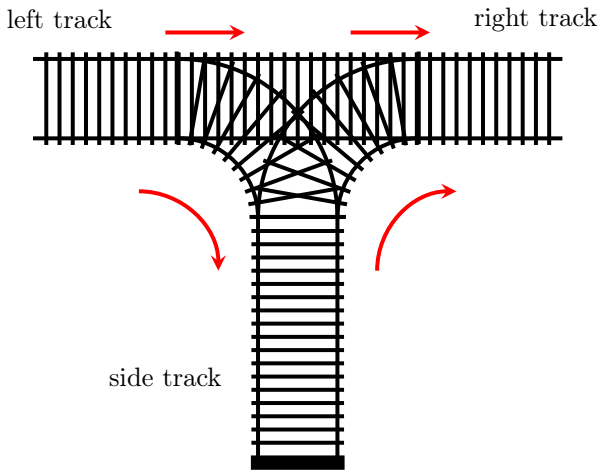

Figure 2. Lausanne Gare

4 *(20 pts)* **Train sequences at Lausanne Gare.** You and your friend are watching the trains at Lausanne Gare. The tracks at Lausanne Gare form a 'T' shape as shown in Fig. 2. You sit at the left end of the tracks and see that n trains labeled $1, 2, \ldots, n$ are lined up in increasing order. Then the trains start moving to the right one by one, first train 1, then 2 and so on. Any train may either turn onto the side track (the side track can hold an infinite number of trains) or keep moving right and leave the station. At any point in time a train on the side track can decide to leave the station through the right track if it's not blocked in. However, the trains can never go back to the left track.

Your friend sits at the right end of the tracks, observes the trains leaving one by one and writes down their numbers in the order that they leave. Your task is to develop an efficient algorithm that decides, given your friend's notes, if that order of trains is possible or if your friend had made a mistake.

Input:	A permutation $a_1, a_2, \ldots, a_n$ of integers between 1 and n.
Output:	**YES** if there exists a way for trains $1, 2, \ldots, n$ to leave Lausanne Gare in this order, and **NO** otherwise.

For example, suppose there are $n = 3$ trains. Then order $2, 1, 3$ (i.e., $a_1 = 2, a_2 = 1, a_3 = 3$) is possible: the first trains turns onto the side track, the second train leaves, the first train exits the side track and leaves and finally the third train leaves as well. However, the order $3, 1, 2$ is not possible.

Design an efficient algorithm for this problem and **analyze its runtime**.

Solution:

(Solution to problem 4 continued)