ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Midterm Exam, Algorithms 2013-2014

- 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 so that a fellow student can understand it. For example, a description of an algorithm should be so that a fellow student can easily implement the algorithm following the description.

Good luck!

Name: _____ **N° Sciper:** _____

Exercise 1 / 15 points	Exercise 2 / 25 points	Exercise 3 / 15 points	Exercise 4 / 20 points	Exercise 5 / 25 points

Total / 100

1 *(15pts)* **Asymptotics.**

1a *(5pts)* Arrange the following functions in increasing order according to asymptotic growth.

$$4 \cdot 16^{n/2}, \quad 10 \log^{3/2} n, \quad n^5, \sqrt{n}, \quad 10 \log n, \quad 100 \cdot 2^n, \quad n^4 \log^2 n, \quad 2 \cdot 3^n$$

1b *(10 pts)* Give tight asymptotic bounds for the following recurrences (assuming that $T(1) = \Theta(1)$):

(i) $T(n) = 3T(n/3) + \Theta(1)$

(ii) $T(n) = 3T(n/3) + \Theta(n)$

(iii) $T(n) = 3T(n/3) + \Theta(n^2)$

(iv) $T(n) = T(n/5) + 2T(2n/5) + \Theta(n)$

(v) $T(n) = 25T(n/5) + \Theta(n^2)$

2 *(25pts)* **Divide and Conquer.** Consider the following procedure UNKNOWN that takes as input an array $A[\ell \dots r]$ of $r - \ell + 1$ numbers with the left-index ℓ and the right-index r:

```
UNKNOWN(A, ℓ, r)
1. if ℓ > r
2.       return −∞
3. else if ℓ = r
4.       return A[ℓ]
3. else
4.       q ← ℓ + ⌊(r−ℓ)/3⌋
5.       ansL ← UNKNOWN(A, ℓ, q)
6.       ansR ← UNKNOWN(A, q + 1, r)
7.       if ansL > ansR
8.             return ansL
9.       else
10.            return ansR
```

2a *(5pts)* Let $A[1 \dots 8] =$ | 6 | 4 | 2 | 9 | 2 | 8 | 7 | 5 |. What does a call to UNKNOWN(A, 1, 8) return?

2b *(8pts)* Let $T(n)$ be the time it takes to invoke UNKNOWN(A, ℓ, r) where $n = r - \ell + 1$ is the number of elements in the array. Give the recurrence relation of $T(n)$.

2c *(12pts) Prove* tight asymptotic bounds of $T(n)$ using the substitution method, i.e., show that $T(n) = \Theta(f(n))$ for some function f.

Solution of 2c: The function $T(n)$ is:

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + 1.$$

We guess that the function $T(n) = \Theta(n)$ and we prove this using the substitution method. First, we prove an upper bound on $T(n)$. We assume the following inductive hypothesis for all $m < n$:

$$T(m) \leq c_1 m - 1$$

for some constant $c_1 > 0$. We will prove that the hypothesis holds for n: (We can assume that n is divisible by 3.)

$$\begin{aligned} T(n) &= T(\frac{n}{3}) + T(2\frac{n}{3}) + 1 \\ &\leq c_1\frac{n}{3} - 1 + c_1\frac{2n}{3} - 1 + 1 \\ &= c_1 n - 1. \end{aligned}$$

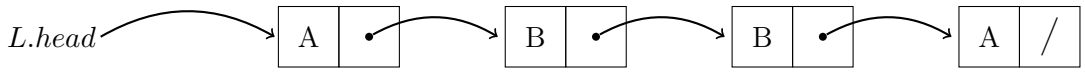
Thus, we have $T(n) = O(n)$.

For the lower bound, we assume that $T(m) \geq c_2 m$ for all $m < n$ and for some constant $c_2 > 0$. Then, we have:

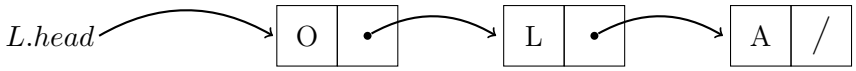
$$\begin{aligned} T(n) &= T(\frac{n}{3}) + T(\frac{2n}{3}) + 1 \\ &\geq c_2\frac{n}{3} + c_2\frac{2n}{3} \\ &= c_2 n. \end{aligned}$$

Thus, we have $T(n) = \Omega(n)$. We can conclude that $T(n) = \Theta(n)$.

3 *(15pts)* **Palindrome.** A word is a palindrome if its reverse is equal to itself. For example, ABBA is a palindrome whereas OLA is not. One way of representing a word in a computer is to have a single-linked list where we have a list-element for each letter. For example, ABBA is represented by the single-linked list



and OLA is represented by the single-linked list



Design and **analyze** an algorithm that, given a pointer to the head of a single-linked list which represents a word, outputs YES if the word is a palindrome and NO otherwise.

For full score your algorithm should run in **linear time** and **should not use any other data structures than single-linked lists**, i.e., no arrays, stacks, queues, etc..

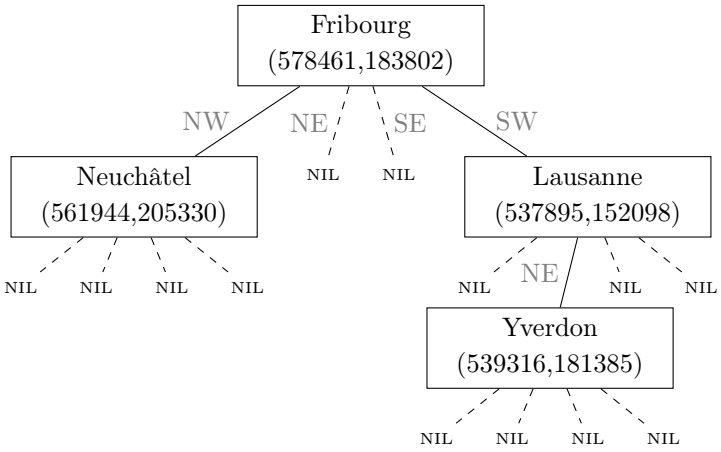


Figure 2. A possible quadtree representation of the cities in Figure 1

4 *(20pts)* **Quadtrees.** A quadtree is a search tree data structure in which each internal node has exactly four children. Quadtrees are usually used to partition the two-dimensional plane by recursively subdividing it into four quadrants or regions. In this exercise, we shall use quadtrees to represent cities according to their geographical position.

Figure 1 shows an example of 4 cities in western Switzerland. It also shows the 4 quadrants induced by the city Fribourg: the first, named NW (for *north-west*), contains the city Neuchâtel; the two following (NE for *north-east* and SE for *south-east*) are empty; the last (SW for *south-west*) contains the two remaining cities, Yverdon and Lausanne.

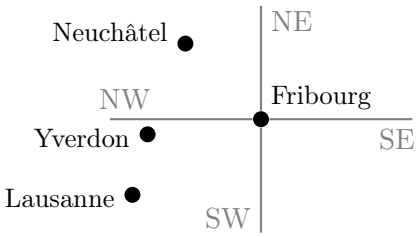


Figure 1. Geographical location of 4 cities in Switzerland

Each node v in the quadtree will store the name ($v.name$) and the coordinates ($v.x$ and $v.y$) of the cities. In addition, the node v will contain a pointer $v.p$ to its parent (or NIL if its the root) and pointers $v.NW, v.NE, v.SE, v.SW$ to its four children. Similar to binary search trees the key properties that make quadtrees useful are the following:

- if u is in the quadtree rooted by $v.NW$ then $u.x < v.x$ and $u.y \geq v.y$;
- if u is in the quadtree rooted by $v.NE$ then $u.x \geq v.x$ and $u.y > v.y$;
- if u is in the quadtree rooted by $v.SE$ then $u.x > v.x$ and $u.y \leq v.y$;
- if u is in the quadtree rooted by $v.SW$ then $u.x \leq v.x$ and $u.y < v.y$.

Figure 2 shows a possible quadtree representation of the cities in Figure 1. The root is the node corresponding to Fribourg that has x and y coordinates equal to 578461 and 183802, respectively.

Design (give the pseudocode) and **analyze** the running times of the following procedures for the quadtree data structure:

(10pts) SEARCH(Q.root, x, y) — prints the name of the city located at (x, y) if such a city exists in the quadtree rooted at $Q.root$.

(10pts) PRINTSOUTHMOST(Q.root) — prints the name of the south most city (the city with the smallest y coordinate) in the quadtree rooted at $Q.root$.

Your runtime analyses should be tight for the typical case when the height of the quadtree is logarithmic in the number of cities that it contains. In addition, for full score, the running times of your implementations should not be unnecessary large.

5 *(25 pts)* **Restaurant placement.** Justin Bieber has surprisingly decided to open a series of restaurants along the highway between Geneva and Bern. The n possible locations are along a straight line, and the distances of these locations from the start of the highway in Geneva are, in kilometers and in arbitrary order, $m_1, m_2, \ldots, m_n$. The constraints are as follows:

- At each location, Justin may open at most one restaurant. The expected profit from opening a restaurant at location i is p_i, where $p_i > 0$ and $i = 1, 2, \ldots, n$.
- Any two restaurants should be at least k kilometers apart, where k is a positive integer.

As Justin is not famous for his algorithmic skills, he needs your help to find an optimal solution, i.e., **design and analyze** an *efficient* algorithm to compute the maximum expected total profit subject to the given constraints.