Teaching assistant : Malory Marin, malory.marin@ens-lyon.fr

Throughout the practical sessions [*travaux dirigés*] of this class, you will **work on exercises** on paper and **implement** some of the algorithms encountered during the lectures.

(a) Make sure you have a working `Python` setup on your computer and you have the libraries `Matplotlib`, `NumPy` and `NetworkX` installed. Having `Jupyer Notebook` will also be useful.

(b) The optional exercises marked with *[*]* are *pour aller plus loin.*

(c) A classic and widely-used reference on algorithms is the "CLRS" first published in 1990:

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms* (4th ed.), MIT Press and McGraw-Hill, 2022, 1312 pp.

The library (Monod) has several copies. A French translation of the 3rd edition also exists. Exercise 2 is based on the corresponding chapter of this book.

(d) Many of these exercises were provided by the previous teaching assistants, Pegah Pournajafi and Kristóf Huszár.

Exercise 1. *Heaps, max-heaps, heapsort* [tri par tas]

The goal of this exercise is to (re)view the binary *heap* data structure, the procedure for turning a heap into a *max-heap*, and then extend this procedure to the full Heapsort algorithm.

A *heap* is an array $A[1...A.\text{length}]$ that we can view as a nearly complete binary tree. The parent of a node i in the tree is $\lfloor i/2 \rfloor$, its left child $2i$ and its right child $2i+1$. A *max-heap* is a heap where the value of each internal node is greater than or equal to the values of its left and right children.

(a) Basic properties of heaps and max-heaps (warm-up)

(i) What are the minimum and maximum numbers of elements in a heap of height h?

(ii) Show that an n-element heap has height $\lfloor \log(n) \rfloor$.

(iii) Show that in any subtree of a max-heap, the root of the subtree contains the largest value occurring anywhere in that subtree.

(iv) Where in a max-heap might the smallest element reside, assuming that all elements are distinct?

(v) Is the array with values $\langle 23; 17; 14; 6; 13; 10; 1; 5; 7; 12 \rangle$ a max-heap?

(vi) Show that, with the array representation for storing an n-element heap, the leaves are the nodes indexed by $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \dots, n$.

(b) From heaps to max-heaps: the Max-Heapify and Build-Max-Heap procedures

Given an array A – which we intend to sort using Heapsort – we consider two attributes:

- *A.length*, to denote the number of elements in A;

- *A.heap-size*, where $0 \leq A.heap\text{-}size \leq A.length$, to indicate that the heap currently consists of the elements in the subarray $A[1 \ldots A.heap\text{-}size]$ of A. The distinction between *A.length* and *A.heap-size* will become relevant in part (c).

The MAX-HEAPIFY procedure can be described in pseudocode as follows:

```
MAX-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heap-size and A[l] > A[i] then
    largest = l
else
    largest = i
end if
if r ≤ A.heap-size and A[r] > A[largest] then
    largest = r
end if
if largest ≠ i then
    exchange A[i] with A[largest]
    MAX-HEAPIFY(A, largest)
end if
```

(i) Illustrate the operation of MAX-HEAPIFY$(A, 2)$ on the array $A = \langle 16; 4; 10; 14; 7; 9; 3; 2; 8; 1 \rangle$, where $A.heap\text{-}size = 10$.

(ii) Illustrate the operation of MAX-HEAPIFY$(A, 3)$ on the array $A = \langle 27; 17; 3; 16; 13; 10; 1; 5; 7; 12; 4; 8; 9; 0 \rangle$, where $A.heap\text{-}size = 14$.

(iii) *[*]* Starting with MAX-HEAPIFY, write pseudocode for MIN-HEAPIFY(A, i), which performs the corresponding manipulation on a *min-heap*. How does the running time of MIN-HEAPIFY compare to that of MAX-HEAPIFY.

(iv) *[*]* Show that the worst-case running time of MAX-HEAPIFY on a heap of size n is $\Omega(\log n)$. (*Hint:* For a heap with n nodes, give node values that cause MAX-HEAPIFY to be called recursively at every node on a simple path from the root down to a leaf.)

```
BUILD-MAX-HEAP(A)
1: A.heap-size = A.length
2: for i = ⌊A.length/2⌋ downto 1 do
3:     MAX-HEAPIFY(A, i)
4: end for
```

(v) Illustrate the operation of BUILD-MAX-HEAP on the array $A = \langle 4; 1; 3; 2; 16; 9; 10; 14; 8; 7 \rangle$

(vi) *[*]* Why do we want the loop index i in line two of BUILD-MAX-HEAP to decrease from $\lfloor A.length/2 \rfloor$ to 1 rather than increase from 1 to $\lfloor A.length/2 \rfloor$?

(c) Heapsort(A)

```
Build-Max-Heap(A)
for i = A.length downto 2 do
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    Max-Heapify(A, 1)
end for
```

(i) Illustrate the Heapsort on the array $A = \langle 16; 14; 10; 8; 7; 9; 3; 2; 4; 1 \rangle$.

Exercise 2. ***Euclidean minimum spanning tree***

In this exercise you will compute and plot the *Euclidean minimum spanning tree* of a large random planar point set by using the `Python` libraries `Matplotlib`, `NumPy`, and `NetworkX`.

You are encouraged to use the resources available on the Internet, especially the documentation of the libraries mentioned above.

(a) Initialize an integer variable n.

(b) Sample n random points from within a square on the Euclidean space. Let's call the resulting point set P.

(c) Draw a scatter plot of P.

(d) Compute the *distance matrix* of P, i.e., an n-by-n matrix D, where $D_{i,j}$ is the Euclidean distance between the points i and j. (Note that the resulting matrix is symmetric and has only zeros on its diagonal.)

(e) Using the functions of `NetworkX`, generate an edge-weighted graph $G = (V, E)$, where $|V| = n$ and the weight of the edge $\{i, j\}$ equals $D_{i,j}$.

(f) Using the appropriate function of `NetworkX`, compute a minimal spanning tree T of G.

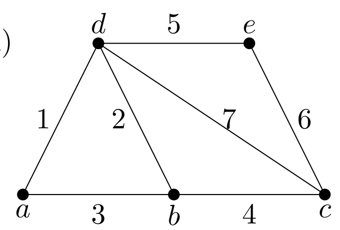
(g) Produce a geometrically faithful drawing of T.

(h) What do you see on the drawing? Do you see pairs of edges that are crossing? Why?

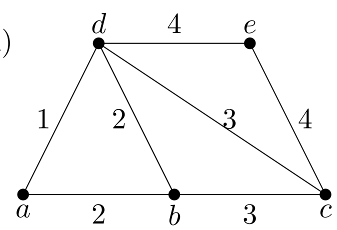
Exercise 3. ***Minimum spanning trees***

(a) Use Kruskal's algorithm to find minimum spanning trees (MSTs) of the following graphs:

(i)



(ii)



(b) How many MSTs do the above graphs have?

(c) (i) Prove that a tree on n vertices has exactly $n - 1$ edges.

(ii) Let $G = (V, E)$ be a connected graph with $|V| = n$ and $|E| = n - 1$. Is it true that such a G is always a tree?

(d) Let G be a connected edge-weighted graph.

(i) Suppose T is an MST of G. Show that for any edge e of G which is not in T , $T \cup \{e\}$ has exactly one cycle. Moreover, show that e has the maximum weight among all the edges in this unique cycle.

(ii) Let e be an edge in G whose weight is strictly smaller than the weight of any other edge in G. Show that e belongs to every minimum spanning tree of G.

(iii) *[*]* Show that if all the edge-weights of G are distinct, then G has a unique MST.