INF539: programming exercises

F. MORAIN

September 16, 2020– version 2200

1 Arrays and lists

a) Program the following variant of Eratosthenes's sieve:

Sieve:
1. Set $T[x] = 1$ for $x \in [2, X]$;
2. **for** $p = 2$ **to** $\lfloor \sqrt{X} \rfloor$ **do**
   if $T[p] = \emptyset$ **then** (**p is prime* *)
   2.1 $x := 2p$;
   2.2 **while** $x \leq X$ **do** (**remove multiples of p* *)
   2.2.1 $T[x] := 0$;
   2.2.2 $x := x + p$.

Postsieve: print all $x \in [2..X]$ s.t. $T[x] = 1$.

Use the type `char` for `T`. Try $X = 10^6$, $X = 10^8$. How many primes do you find?

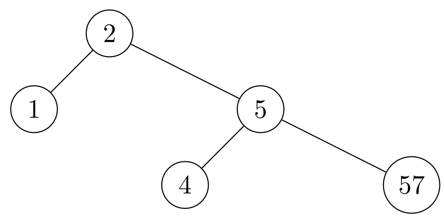
b) We can replace 2.1 by
2.1 $x := p^2$;
(why?) and 2.2.2 by
2.2.2 $x := x + 2p$.
(why?). Improve the postsieving and program this variant. Be sure to recover the same values as in a).

c) [optional] Show how to use 1 bit per value of x in T and program the variant. At the same time, we don't care about even values in $[2, X]$. Same verification.

d) We come back to the original algorithm where we want the prime factors of any x in T. Use the given implementation for lists to program the original variant. As a matter of fact, T will be an array of lists, initialized to `NULL`.

2 Trees

The aim of this exercise is to program *binary search trees.* A tree $T = (r, T_l, T_r)$ is a binary search tree (BST) if all nodes in T_l are $\leq r$ and all nodes of T_r are $> r$, the same property being satisfied by T_l and T_r. The following Figure gives an example of such a BST.



Inserting x in a BST T means putting x somewhere in T while respecting the property. The procedure is

- If T is empty, create a new tree $(x, \emptyset, \emptyset)$ which satisfies the property.
- If $T = (r, T_l, T_r)$, compare x with r and insert x in T_l or T_r depending on the answer.

a) Draw a picture corresponding to the insertion of the elements in `t` one by one.

```
int t[11] = {2, 5, 4, 57, 4, 95, 87, 1, 67, 96, 91};
```

b) Program the insertion in a BST.

c) Searching in a BST uses the same principle. Program this.

d) Write a program that counts the number of times an integer is present in a given file. Try it on the file `integers.in` .

e) Suppose T has n nodes. What is the average cost of all operations?

f) [optional] Program suppression of a given node in a BST.

3 Hashing

Write a program that counts the number of times an integer is present in a given file using carefully chosen hash table and hash function. Try it on the file `integers.in` .

4 Graphs

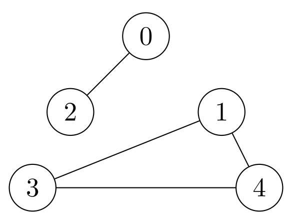
a) Program a DFS that enumerates all connected components of a unoriented graph G, taken from a file. The format will be

```
n m
0 k v00 v01 ... v0k
...
```

where `n` is the number of vertices labelled in $[0..n[$, m the number of edges. Follow `n` lines containing the index of a vertex i, the number of neighbours of i followed by a list of the indices of the neighbours. For instance:

```
5 6
0 1 2
1 0
2 0
3 1 1
4 2 1 3
```

corresponding to the graph with two connected components:



The first task is to add an edge (i, j) each time you encounter (j, i).

b) Generate random graphs by fixing some value of n and varying m from n to n^2 printing the number of connected components appearing. What happens?