

## VE477

### Introduction to Algorithms

#### Lab 7

Manuel — UM-JI (Fall 2017)

#### Goals of the lab

- Search algorithms
- Complexity vs. running time
- Python Programming

*Unless specified otherwise, all the programs are expected to be completed in Python.*

In this lab we want to compare three search algorithms. Let  $A$  be an array of size  $n$  and  $k$  be a value to find in  $A$ .

1. The first algorithm, `RandomSearch`, consists in randomly searching  $A$  for  $k$ . One simply selects a random index  $i$  and test if  $A[i] = k$ . If `True` the algorithm returns  $i$  and otherwise randomly chooses a new  $i$  until either all the indices have been explored or  $k$  has been found. In this algorithm a same index can be generated more than once.
  - (a) Implement `RandomSearch`.
  - (b) Determine the average number of indices that are picked if:
    - i. There is no index  $i$  such that  $A[i] = k$ .
    - ii. There is exactly one index  $i$  such that  $A[i] = k$ .
    - iii. There is more than one index  $i$  such that  $A[i] = k$ .
2. The second algorithm, `LinearSearch`, consists in performing a linear search on  $A$ , i.e. testing if  $k$  is in  $A[1], \dots, A[n]$ , and either return  $i$ , the index where  $k$  is stored in  $A$ , or exit if  $k$  is not found.
  - (a) Implement `LinearSearch`.
  - (b) Determine the average-case and worst-case running times if:
    - i. There is no index  $i$  such that  $A[i] = k$ .
    - ii. There is exactly one index  $i$  such that  $A[i] = k$ .
    - iii. There is more than one index  $i$  such that  $A[i] = k$ .
3. The third algorithm, `ScrambleSearch`, consists in randomly permuting  $A$  into  $A'$  and then run the `LinearSearch` algorithm on  $A'$ .
  - (a) Implement `ScrambleSearch`.
  - (b) Determine average-case and worst-case running times if:
    - i. There is no index  $i$  such that  $A[i] = k$ .
    - ii. There is exactly one index  $i$  such that  $A[i] = k$ .
    - iii. There is more than one index  $i$  such that  $A[i] = k$ .
4. From the previous complexities which algorithm seems to be the best.
5. Generate  $A$ , a random array of one million elements, and time each of the previous algorithms searching a random  $k$  on it. Repeat the process one thousand times, keeping track of the previous runs.
  - (a) Which algorithm performs best in practice?
  - (b) Discuss the result by comparing to your expected result in item 4.