Randomized Algorithms (RA-MIRI): Assignment #3

The goal of this assignment is that you carry out a detailed empirical study of the performance of skip lists. More specifically, of the average performance of searches and of memory requirements, as a function of the number n of elements in the skip list and of the parameter q.

You will need first to program a `Set` class using skip lists. C++ is the preferred choice, but other imperative programming languages are also fine, as long as the code is well documented and good programming practices (e.g., modularization) are applied. Make sure that the user can set the relevant parameter q. For instace, in C++, you might have

```
template <class T>
 class Set {
    public:
    Set(double q); // constructor, creates an empty Set, sets the
                          // skip list parameter q to the given value
    ~Set();             // destructor
    Set(const Set& S); // copy constructor
    Set& operator=(const Set& S); // assigment
    void insert(const T& x); // inserts x in the set
    void remove(const T& x); // removes x from the set, if present
    bool contains(const T& x) const; // returns true iff x is in the set
    ...
   };
```

Use the convention that

$$\Pr\{\text{height}(x) = i\} = pq^{i-1}, \qquad p = 1 - q$$

for any item x and any $i > 0$. Ideally, the implemented class should be generic (`Set<T>`, as in the example above, for any type `T` which has defined a total order), but if not, the class should support search, insertion and deletion in a finite set of integers.

For the experimental study you should use large sets of n elements. Without loss of generality, the set can be the numbers $\{1, \ldots, n\}$. For any fixed n and q, generate M different skip lists (for instance, $M = 100$) and measure for each one the total search cost and the total memory consumption. Average these quantities over all M instances to get estimates $C_{n,q}$ and $S_{n,q}$ of the expected total search cost and expected memory, respectively. To compare these quantities with their theoretical values use for the expected total cost

$$Qn \log_Q n + n\left(\frac{Q}{L}(\gamma - 1) + \frac{1}{L} - \frac{Q}{2}\right),$$

where $Q = 1/q$, $L = \ln Q$ and $\gamma = 0.5772156649$, and for the expected memory usage (=number of pointers) use

$$n/p + \log_Q n.$$

Your class should support these two methods (I use C++ syntax as before):

```
template <class T>
  class Set {
    public:
    ...
    int total_search_cost() const;
    int number_pointers() const;
    ...
  };
```

that report the total search cost and the number of pointers used by the particular instance of skip list on which they are applied. For the number of pointers, sum the height (= number of pointers) of all nodes in the skip list, including the header. Alternatively, the insertion and deletion methods may maintain this quantity updated at all times. For the total search cost the most efficient algorithm should know the size of each "subskiplist"; however, you can use the trick described below, since we only consider skip lists that store the numbers 1 to n. Suppose the skip list S is of height h and let k be the first element of height h in the skip list. Then the total search cost is n plus the total search cost of the sub skiplist σ with the elements 1 to $k-1$ (of size $k-1$) plus the total search cost of the sub skiplist τ with elements $k+1$ to n (of size $n-k$). A recursive implementation going down one level to compute the total search cost of σ, and following the horizontal pointer to the k-th node to compute the total search cost of τ, while keeping track of the size of the current sub skip list, is easy to implement and has cost $\Theta(n)$.

Run the experiments for different values of n, for example, from $n = 2000$ to $n = 20000$ in steps of 100, and for different values of q, for example, $q = 0.1, 0.2, \ldots, 0.9$.

Once the full suite of experiments has been executed and data has been gathered, you have to prepare a report.

1. Describe briefly your implementation of skip lists and the program to execute the experiments. Give full listings of the code as an appendix of your report.

2. Describe briefly the experimental setup, how many different combinations of the parameters n and q have you studied, how many runs M have you performed for a particular n and q.

3. Provide tables and plots summarizing the results of the experiments. In particular, you should give plots showing how $C_{n,q}$ varies with n, and how it varies with q (to study the variation with q, it is useful to plot $C_{n,q}/(n \ln n)$ instead of $C_{n,q}$). Do the same with $S_{n,q}$. Avoid 3-D plots.

4. Compare the experimental results with the theoretical predictions. Plots combining the theoretical values and the experimental results are useful,

but it is also important to quantify the difference between the theoretically predicted values and the empirical estimations; in particular, you should obtain the standard deviation of the estimations of $C_{n,q}$ and $S_{n,q}$ that you get from the average values.

5. Write down your conclusions.

It is not mandatory, but it will be positively evaluated if you also conduct an empirical study relating $C_{n,q}$ and the actual execution time of searches in the skip list, following similar steps as described above.

Instructions to deliver your work

Submit your work using the FIB-Racó. The deadline for submission is December 11th, 2022 at 23:59. It must consist of a zip or tar file containing all your source code, auxiliary files and your report in PDF format. Include a README file that briefly describes the contents of the zip/tar file and gives instructions on how to produce the executable program(s) used and how to reproduce the experiments. The PDF file with your report must be called `YourLastName_YourFirstName-3.pdf`,

N.B. I encourage you to use LaTeX to prepare your report. For the plots you can use any of the multiple packages that LaTeX has (in particular, the bundle TikZ+PGF) or use independent software such as matplotlib and then include the images/PDF plots thus generated into your document.