Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Übungen zur Vorlesung Informatik II (D-BAUG) FS 2017

D. Sidler, F. Friedrich

http://lec.inf.ethz.ch/baug/informatik2/2017

Problem set # 5 **20.3.2017 – 28.3.2017**

Problem 5.1. Compare sort algorithms.

Consider an array $A[1 \dots n]$ and the following (not necessarily optimal) Java implementations of the sort algorithms *Bubblesort*, *Insertion sort*, *Selection sort* and *Quicksort*.

The functions are called with $l = 1$ and $r = n$ to sort A in ascending order.

```
void bubbleSort(int[] A, int l, int r) {
   for(int i=r; i>l; i--)
      for(int j=l; j<i; j++)
         if(A[j] > A[j+1])
            swap(A, j, j+1);
}

void insertionSort(int[] A, int l, int r) {
   for(int i=l; i<=r; i++)
      for(int j=i-1; j>=l && A[j] > A[j+1]; j--)
         swap(A, j, j+1);
}

void selectionSort(int[] A, int l, int r) {
   for(int i=l; i<r; i++) {
      int minJ = i;
      for(int j=i+1; j<=r; j++){
         if(A[j] < A[minJ])
            minJ = j;
      }
      if(minJ != i)
        swap(A, i, minJ);
   }
}

void quicksort(int[] A, int l, int r) {
   if(l<r){
      int i=l+1, j=r;
      do{
         while(i<j && A[i] <= A[l])
            i++;
         while(i<=j && A[j] >= A[l])
            j--;
         if(i<j) swap(A, i, j);
      } while(i<j);
      swap(A, l, j);
      quicksort(A, l, j-1);
      quicksort(A, j+1, r);
   }
}
```

The function `swap(A, i, j)` exchanges the elements of A at positions i and j. Provide for each algorithm an asymptotic lower and upper bound on the number of calls to $swap$. Also give the sequence of integers $1, 2, \dots, n$ where these cases occur. Describe the sequence generically depending on n. For instance, describe the sequence of descending numbers as $n, n-1, \dots, 1$.

Submission link: https://codeboard.ethz.ch/inf2baugex05t01

Problem 5.2. Exceptions

Open the code template at: https://codeboard.ethz.ch/inf2baugex05t02.

Your task is to open the file at the location "./Root/gedicht.txt", read the file and print out the number of lines which are not empty.

For reading the file use the `FileReader` and `BufferedReader` provided by the `java.io` library.

To successfully compile the program you have to catch the `IOException`.

Advice: Look up the method `readLine()` of the class `BufferedReader` in the Java 8 API documentation[1]. To check if a string is empty use the function `isEmpty()` function.

To test your program un-comment the annotation `@RunTests`. Once you pass the test you can submit your program.

Problem 5.3. Statistics on a Stream of Data ("Online Algorithm")

Open the code template at: https://codeboard.ethz.ch/inf2baugex05t03a.

Assume you have a continuous stream of data (we simulate this with user input). For each incoming data point of type double, you are asked to compute the mean $\mu_n = \sum_{i=1}^{n} \frac{x_i}{n}$ and sum of squares $\sigma_n^2 = \sum_{i=1}^{n}(x_i - \mu_n)^2$ of the data received until now.

Make sure your algorithm can do this in $\mathcal{O}(1)$ for each new data point arriving. Hint: think about how to compute μ_{n+1} from μ_n and x_{n+1} (write down the sum formula) and try to do the same for σ_{n+1}^2.

Once you have implemented the functionality, you can test your program by un-commenting the annotation `@RunTests`. Once you pass the test you can submit your program.

Question: can you implement a similar algorithm for the median? If so, how? If not, why not?

Provide your answer here: https://codeboard.ethz.ch/inf2baugex05t03b

[1] http://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html