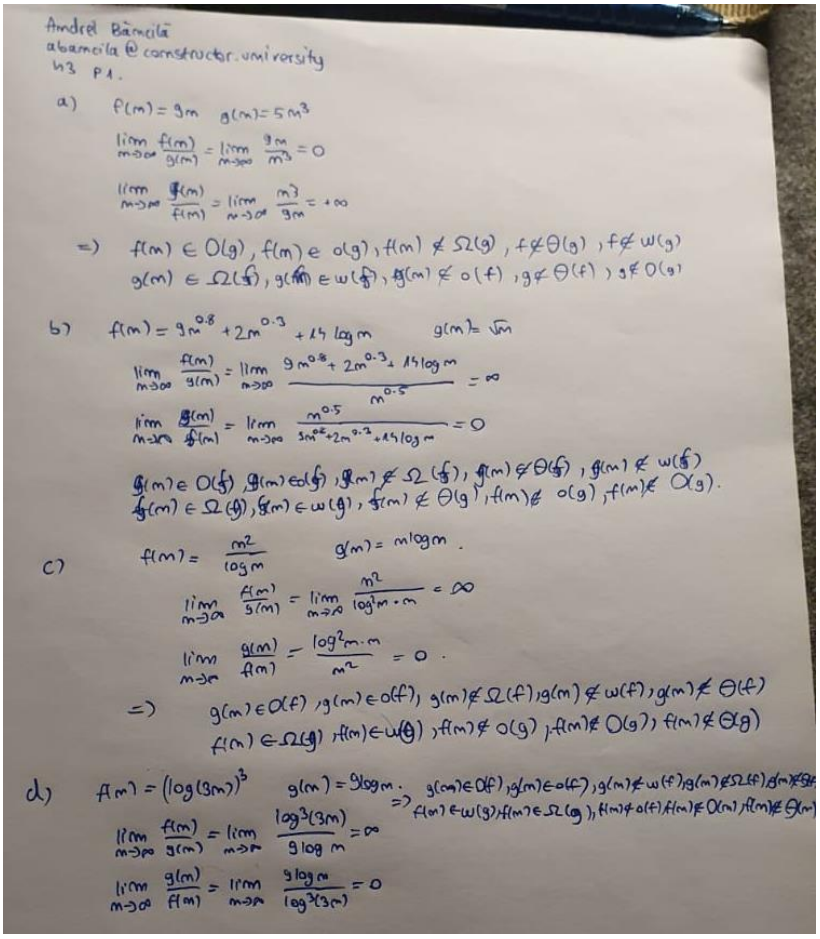
21.02.2023
Spring 2023

Andrei Bancila
abancila@constructor.university

Algorithms & Data Structures
Homework #3

Problem 3.1 *Asymptotic Analysis*



Problem 3.2 *Selection Sort*

a)

Andrei Bancila
abancila@constructor.university

```cpp
/*
    CH-231-A
    h3_p2.cpp
    Andrei Bancila
    abancila@constructor.university
*/
#include <iostream>
#include <vector>

using namespace std;

void fastSwap(int &a,int &b){a = a ^ b;b = a ^ b;a = a ^ b;}

void selectionSort(vector<int> &v)
{
    int n = v.size();
    for(int i = 0 ; i < n-1 ; i++)
    {
        int mi = i;
        for(int j = i+1 ; j < n ; j++)
            if(v[mi] > v[j])
                mi = j;
        if(mi != i)
            fastSwap(v[mi], v[i]);
    }
}

vector<int>v;

int main()
{
    int n; cin >> n;
    for(int i = 1 ; i <= n ; i++)
        {
            int x;
            cin >> x;
            v.push_back(x);
        }
    selectionSort(v);
    for(auto it : v) cout << it << ' ';
    return 0;
}
```

21.02.2023
Spring 2023

Andrei Bancila
abancila@constructor.university

b)

Invariant: v[0:i-1] contains the first i smallest elements in sorted order and mi always contains the position of the minimum element in range [i:j]

Initialization: at first, the subarray v[0:i-1] is empty and sorted

Maintenance: All the elements from [0:i-1] will be sorted and smaller than the elements from [i:n-1]

The inner loop will find the minimum of the unsorted array and swap it with element at position i. Therefore the element on the i th position will be bigger than all the elements of the left of it and smaller than all elements on the right of it. On the next iteration, the sorted array will have the range [0:i] and so on…

Termination At the end of the for (outer loop) everything stops. The maintenance held so we have the following special case for the invariants: the array from 0 to n-1 is sorted and the element of position n-1 is smallest of the elements [n-1:n-1], which is true. That means that the array is sorted.

c)

Case A: The greatest number of swaps of Selection Sort is n-1, and it is achieved in a special type of sequence, namely an alternating increasing and decreasing sequence.

For instance, sequences like 8 7 9 6 10 5 11 4, n = 8 will have n-1 swaps

So basically, all sequences like a1 b1 a2 b2 a3 b3 … work where bk < bk-1 < … < b1 < a1 < a2 < … < am, where k and m are the sizes of sequences a and b

To generate this case, we use random numbers in ranges to generate 2 sequences of size m and k (as described above), then sort them each and merge them into the sequence in question to get the worst case complexity.

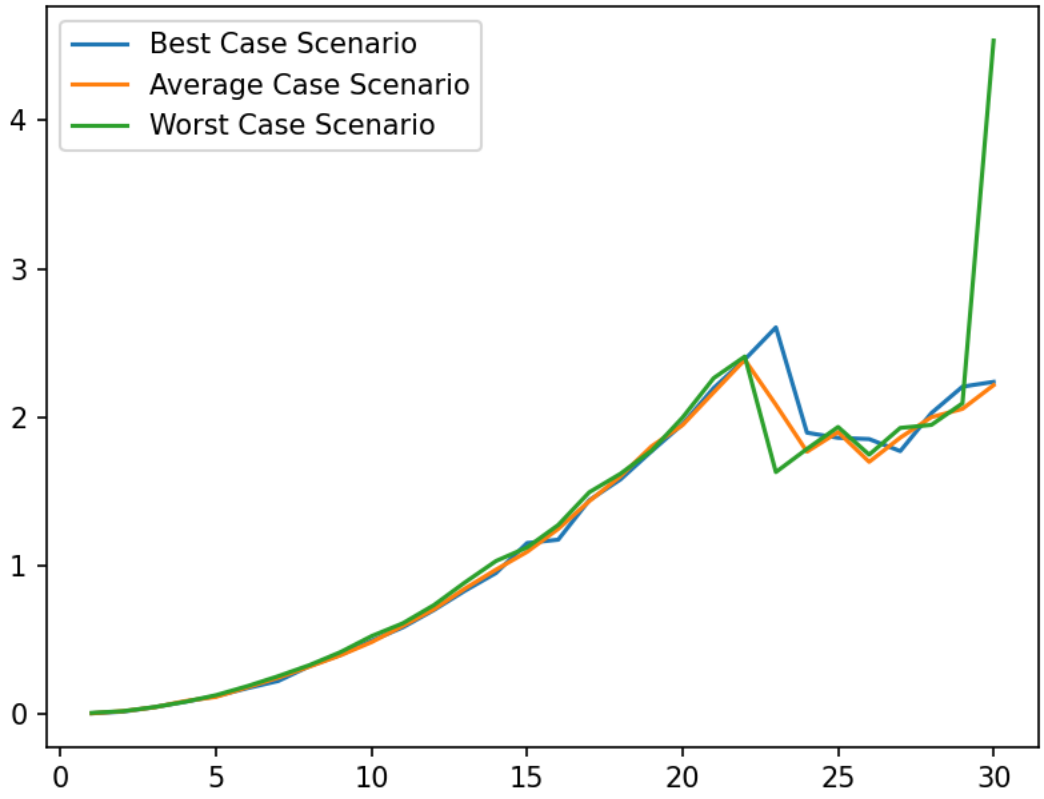
Case B: The best case we can have for Selection Sort is when the array is already sorted, so 0 swaps.

To generate it we use a random number generator for the elements and sort the array beforehand using any type of sorting algorithm.

d)

21.02.2023
Spring 2023

Andrei Bancila
abancila@constructor.university



The numbers on the Y axis are seconds and the numbers on the x axis are the sizes of the generated sequences of numbers (in thousands)