Name:

Name:

Algorithms

CS Fundamentals UVMSC-INF101A

Laboratory 2: Complexity of sorting algorithms

Télécom Bretagne – Département Image et Traitement de l'Information
October 2018

Q1. Propose a strategy or test plan to verify that the three sorting algorithms work correctly and describe how it was applied. Counting results are not required at this point.

We used a simple debugging strategy by printing before and after the arrays to sort the arrays with a small set of random integers. And also used it more detailed to understand the different steps of exchange in the array.

Q2. Indicate where and which counters were applied in the selection sort code.

```
for (i = 0; i < size - 1; i ++) {
    min = i;
    complexite->nb_copies ++;
    for (j = i + 1; j <= size - 1; j ++) {
      if(data[j] < data[min]) {
         min = j;
        complexite->nb_copies ++;
        complexite->nb_comparaisons ++;
      }
      if(min != i) {
         echanger(data, min, i);
        min = i;
        complexite->nb_echanges ++;
        complexite->nb_copies += 4;
        complexite->nb_comparaisons ++;
      }
    }
  }
```

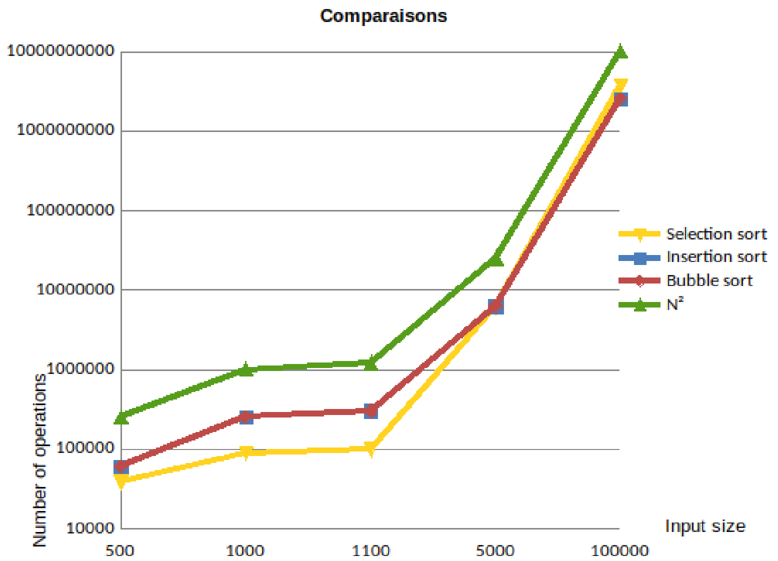
Q3. Indicate where and which counters were applied in the insertion sort code.

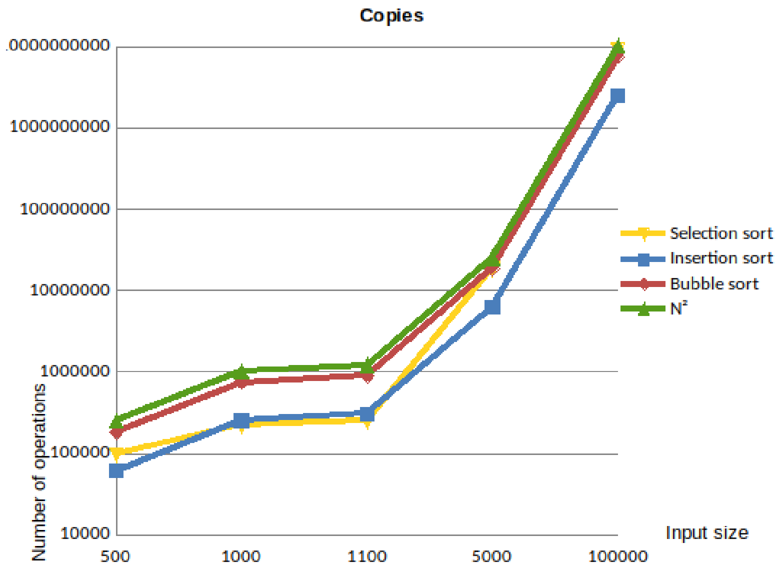
```
for (i = 1; i < size; i ++) {
  v = data[i];
  j = i;
  complexite->nb_copies += 2;
  while (j > 0 && data[j - 1] > v) {
    data[j] = data[j - 1];
    complexite->nb_copies ++;
    complexite->nb_comparaisons ++;
    j --;
  }
  data[j] = v;
  complexite->nb_copies ++;
}
```

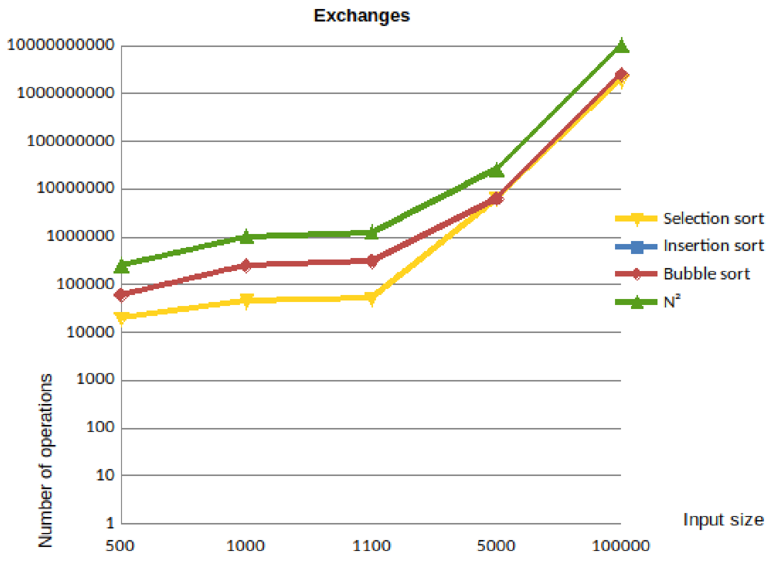
Q4. Indicate where and which counters were applied in the bubble sort code.

```
for (i = size - 1; i >= 1; i --) {
  for (j = 1; j <= i; j ++) {
    if(data[j - 1] > data[j]) {
      echanger(data, j - 1, j);
      complexite->nb_copies += 3;
      complexite->nb_echanges ++;
      complexite->nb_comparaisons ++;
    }
  }
}
```

Q5. Complete the spreadsheet file with the corresponding number of counted instructions, to summarize your results.







Q6. Analyze the operation counting results for each sorting algorithm, according to the relevancy of examined parameters reported in the spreadsheet and the expected complexity. Compare the three algorithms.

1. Comparison: Here the complexity of the insertion and bubble sort algorithm are reacting similarly while selection sort has a favorable tendency with respect to the others but later intersects with an increase in input size.

2. Copies: We can see the same behavior between bubble and insertion sort even if insertion sort is a little bit more favorable. For Selection sort, as in comparison, it has two stages, the first it seems that it has a favorable behavior respecting the others but afterwards it acquires a wort one.

3. Exchange: The only algorithms that participates are bubble and selection sort since insertion sort doesn’t have any exchanges.

4. Expected Tendency: In every graph, the tendency is clearly n^2 as expected (but for insertion sort in the exchange graph).

Comparing the graphs, we can see that all three algorithms have the same behavior, which make us think that they have the same complexity function (and actually they all have O(n^2)). Though, we can realize also that for practical questions, insertion sort is the fastest of the three since it doesn’t apply the exchange operations.