



Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

ALGORYTHMIQUE ET STRUCTURE DE DONNÉES

T1A

RÉSEAU ET SÉCURITÉ

S04 Algo de tri simple

ROTEN MARC

2017/2018

Table des matières

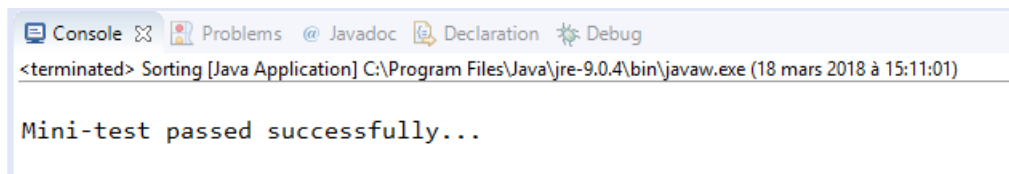
1	Exercices	2
1.1	Selection Sort	2
1.2	ShellSort	2
1.3	Ex 3	3
1.3.1	ShellSort	3
1.3.2	insertion	3
1.3.3	Bubble	3
1.4	Complexité	4
1.4.1	ShellSort	4
1.4.2	insertion	4
1.4.3	Bubble	4
1.4.4	Selection	4
1.5	methode mystère	4
1.6	Bubble sort	4

Chapitre 1

Exercices

1.1 Selection Sort

```
public static void selectionSort(int [] a){
    for(int i=0; i<a.length;i++){
        int minOfTable = i; //we place the 1st elt of our table
                             as our minimum
        for(int j=i+1; j<a.length;j++){//if another elt is
            littler than our min, it becomes the new min
            if(a[j] < a[minOfTable]) { minOfTable = j;}
        }
        if(minOfTable != i){//if the 2 elt are different, we
            swap else no swap
            int tempMin = a[minOfTable];
            a[minOfTable] = a[i];
            a[i] = tempMin;}
    }
}
```



1.2 ShellSort

```
public static void shellSort(int [] a){
    int index=0;
    while(index<a.length) {
        index=3*index+1;
    }
}
```

```

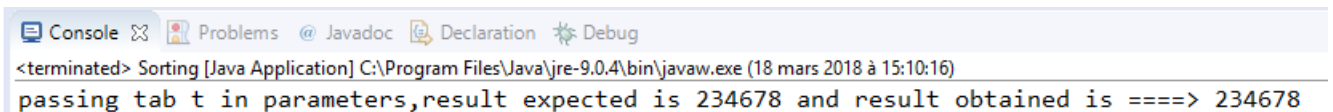
while(index!=0){
    index=index/3;
    for (int i=index;i<a.length;i++){
        int valeur=a[i];
        int j=i;
        while((j>(index-1)) && (a[j-index]>valeur)){
            a[j]=a[j-index];
            j=j-index;
        }
        a[j]=valeur;
    }
}
}

```

```

int [] t = {4, 3, 2, 6, 8, 7};
shellSort(t);
System.out.print("passing tab t in parameters,result
    expected is 234678 and result obtained is ===> ");
for(int i=0;i<t.length;i++) System.out.print(t[i]);

```



The screenshot shows a Java IDE interface with a console window at the bottom. The console output reads: `<terminated> Sorting [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (18 mars 2018 à 15:10:16)` followed by `passing tab t in parameters,result expected is 234678 and result obtained is ===> 234678`.

1.3 Ex 3

Les algos risquant d'être instables sont ceux qui risquent de commuter les elt de mêmes valeur.

1.3.1 ShellSort

Le tri Shell n'est pas stable, risque de commutation dans ce type de tri entre les elt de même valeur

1.3.2 insertion

Cet algo est stable, on compare la valeur de l'elt présent avec son voisin de droite, et si les 2 elt sont équivalents, pas de commutation donc stable

1.3.3 Bubble

Cet algo est stable, j'entend par là qu'on prend la valeur la plus grande et on la pousse jusqu'au fond du tableau, si 2 valeurs équivalentes sont côtes à côtes, pas d'inversion

1.4 Complexité

1.4.1 ShellSort

Cet algo est de complexité $O(n^{1.5})$

1.4.2 insertion

Cet algo est de complexité $O(n^2)$

1.4.3 Bubble

Cet algo est de complexité $O(n^2)$

1.4.4 Selection

Cet algo est de complexité $O(n^2)$

1.5 methode mystère

Cette methode supprime tous les éléments sauf le plus petit de notre liste

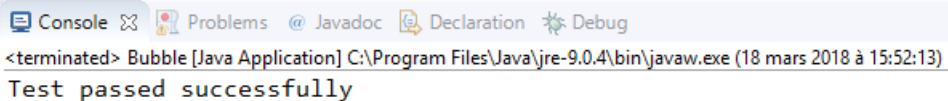
1.6 Bubble sort

```
public class Bubble {
    //-----
    static void bubbleSortList(List l) {
        if (l.isEmpty()) return;
        ListItr li = new ListItr(l);
        boolean goOn=true;
        while(goOn) {
            goOn = false; //we have to reverse for having a non
                           infinite loop here
            for(int i=0;i<l.size();i++) {
                /*
                 * to know if we continue we have to reevaluate goOn
                 * with method bubbleSwapped
                 */
                if(!goOn)goOn=bubbleSwapped(li);
                li.goToNext();
                if(li.isLast())li.goToFirst();// when we finish our
                array, lets do it again
            }
        }
    }
}
```

```

    }
}
}
//-----
//Swaps between left and right element if needed
//Returns true if swap occurred
static boolean bubbleSwapped(ListItr li) {
    /*
     * this method give true if we swapped our 2 elt
     */
    if (li.isFirst() || li.isLast()) return false;
    int before = li.consultAfter();li.goToPrev();
    int after = li.consultAfter();li.goToNext();
    if(before < after) {
        /*
         * we use listIterator and list of S03
         */
        li.removeAfter();
        li.goToPrev();
        li.insertAfter(before);
        li.goToNext();
        return true;
    }
    return false;
}
}

```



```

<terminated> Bubble [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (18 mars 2018 à 15:52:13)
Test passed successfully

```