**Algorithmique / Travaux pratiques**

S04 – Algorithmes de tris simples





05.03.2015

Adriano De Almeida Silva – T-1f

Alex Travasso – T-1f

1. **Tri par sélection**

public static void selectionSort(int[] a) {

// Défini la taille du tableau

int taille = a.length;

// tant qu'il reste des éléments non triés

for (; taille > 1; taille--) {

int max = 0;

int i = 0;

int tmp = 0;

// On cherche le plus grand

while (i < taille) {

if (a[i] > a[max]) {

max = i;

}

i++;

}

// On inverse le plus grand avec le dernier du tableau

tmp = a[taille - 1];

a[taille - 1] = a[max];

a[max] = tmp;

}

//Affiche le tableau trié

for (int j2 = 0; j2 < a.length; j2++) {

System.*out*.println(a[j2]);

}

}

1. **Tri de Shell**

public static void shellSort(int[] a) {

int longueur = a.length;

int k = 0;

while (k < longueur) {

k = 3 \* k + 1;

}

while (k != 0) {

k = k / 3;

for (int i = k; i < longueur; i++) {

int valeur = a[i];

int j = i;

while ((j > (k - 1)) && (a[j - k] > valeur)) {

a[j] = a[j - k];

j = j - k;

}

a[j] = valeur;

}

}

}

Tri par sélection : n’est pas un tri stable, l’ordre d’apparition des éléments égaux n’est pas respecté.

Tri de Shell : n’est pas un tri stable, l’ordre d’apparition des éléments égaux n’est pas respecté.

Tri par insertion : est un tri stable, l’ordre d’apparition des éléments égaux est respecté

Tri à bulles : est un tri stable, l’ordre d’apparition des éléments égaux est respecté



Tri par sélection : O(n^2)

Tri de Shell : O(n^2) à O(n log^2 n) selon l’espacement à chaque étape

Tri par insertion : O(n^2) dans le pire des cas

Tri à bulles : O(n^2)

1. **Explication de la méthode f() :**

Cette méthode va parcourir la liste afin de garder en mémoire (variable m) la plus petite valeur que contient la liste. Ensuite, elle va supprimer cette valeur de la liste, placer l’itérateur au début de la liste et retourner la variable m qui contient la plus petite valeur.

1. **Voici la méthode demandée :**

static void bubbleSortList(List l) {

if (l.isEmpty())

return;

ListItr li = new ListItr(l);

boolean goOn = true;

while (goOn) {

if (bubbleSwapped(li))

li.goToFirst();

else {

li.goToNext();

if (li.isLast())

goOn = false;

}

}

}

// ---------------------------------------------

// Swaps between left and right element if needed

// Returns true if swap occurred

static boolean bubbleSwapped(ListItr li) {

if (li.isFirst() || li.isLast())

return false;

li.goToPrev();

int temp = li.consultAfter();

li.goToNext();

if (temp > li.consultAfter()) {

li.goToPrev();

li.removeAfter();

li.goToNext();

li.insertAfter(temp);

return true;

}

return false;

}