Blin Sébastien
Collin Pierre-Henri

ECOLE SUPERIEURE
D'INGENIEURS DE RENNES

École supérieure d'ingénieurs de Rennes

1ère Année
Parcours Informatique

Algorithmie et complexité

Compte-Rendu TP1

Sous l'encadrement de :

Ridoux Olivier
Maurel Pierre

1 Objectif

Ce premier TP d'Algorithmie et Complexité a pour objectif de comparer 2 algorithmes de tri de 2 complexités différentes. Le premier (tri par insertion) a en effet une complexité théorique en $O(n^2)$, contre $O(nlogn)$ pour le second tri étudié (tri fusion). La dernière partie du TP consistait à examiner un programme et à regarder sa complexité.

2 Moyens mis en œuvre

Pour réaliser ce TP, nous avions à disposition un fichier JAVA à compléter qui s'occupait de l'implementation des tris ainsi que la mesure du temps de calcul. Puis un programme Matlab pour l'affichage des graphes de résultats.
Voici notre implémentation du tri insertion :

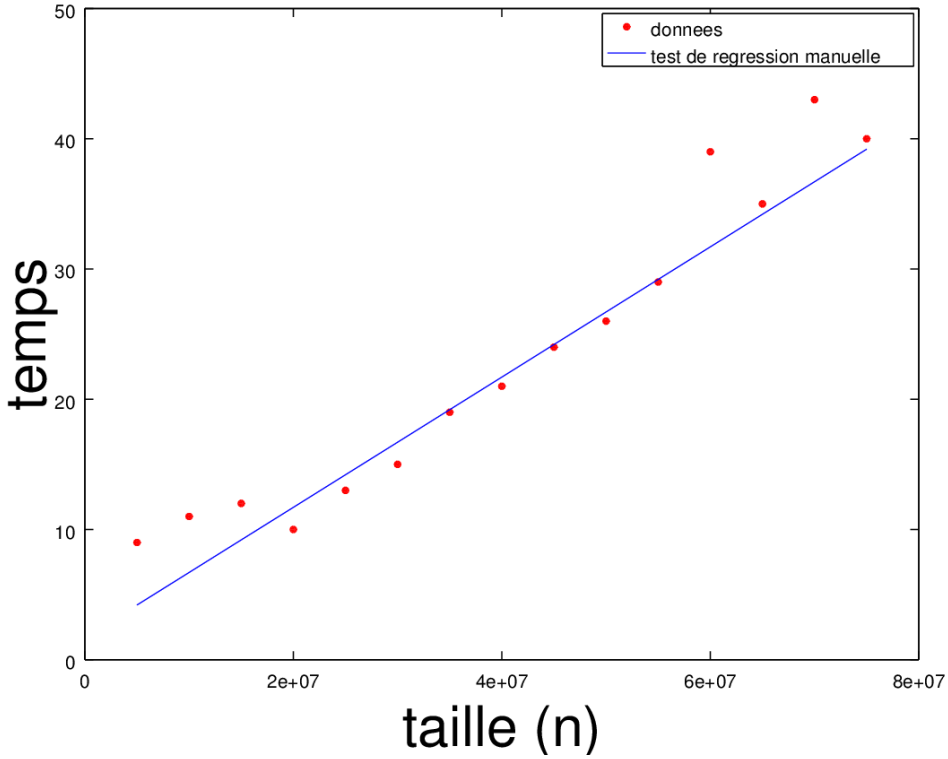
```
public static void triInsertion(int[] t){
    for (int i = 1; i < t.length; i++) {
      int valueToSort = t[i];
      int j = 0;
      for(j = i; j > 0 && t[j - 1] > valueToSort; --j) {
        t[j] = t[j - 1];
      }
      t[j] = valueToSort;
    }
  }
```

Et le tri fusion :

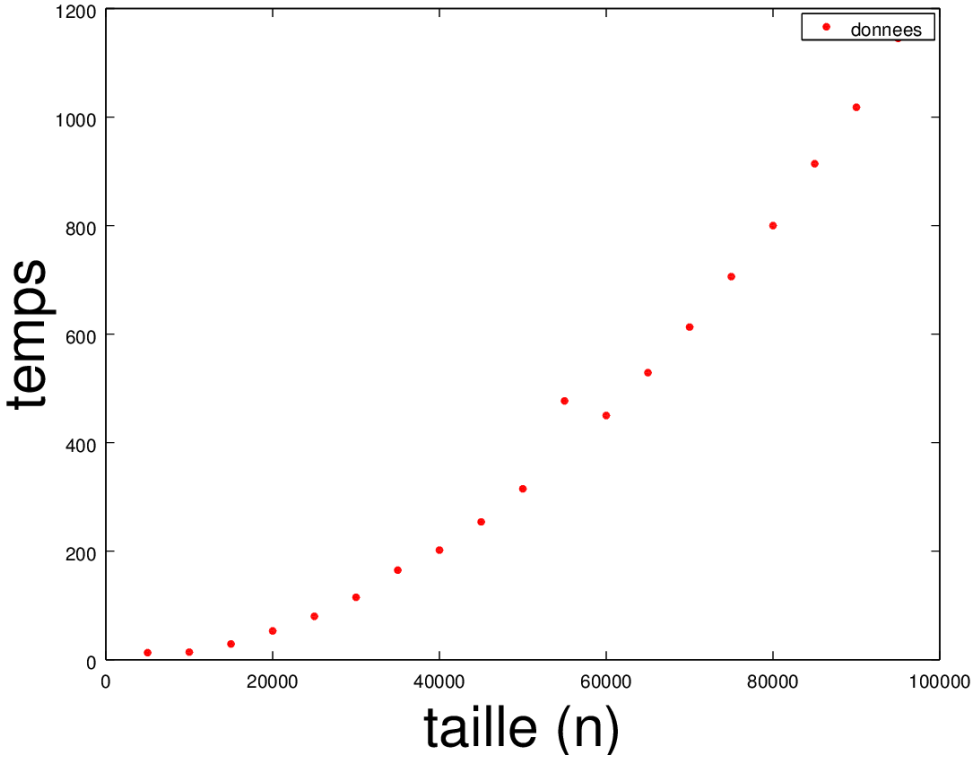
```
/* Sous-fonction (recursive) pour le tri fusion
   * Trie le sous-tableau t[debut]..t[fin]
   */
private static void triFusion(int[] t, int debut, int fin){
  if(debut < fin)
  {
    int milieu = (debut+fin)/2;
    triFusion(t, debut, milieu);
    triFusion(t, milieu+1, fin);
    fusionner(t, debut, milieu, fin);
  }
}
```

3 Résultats

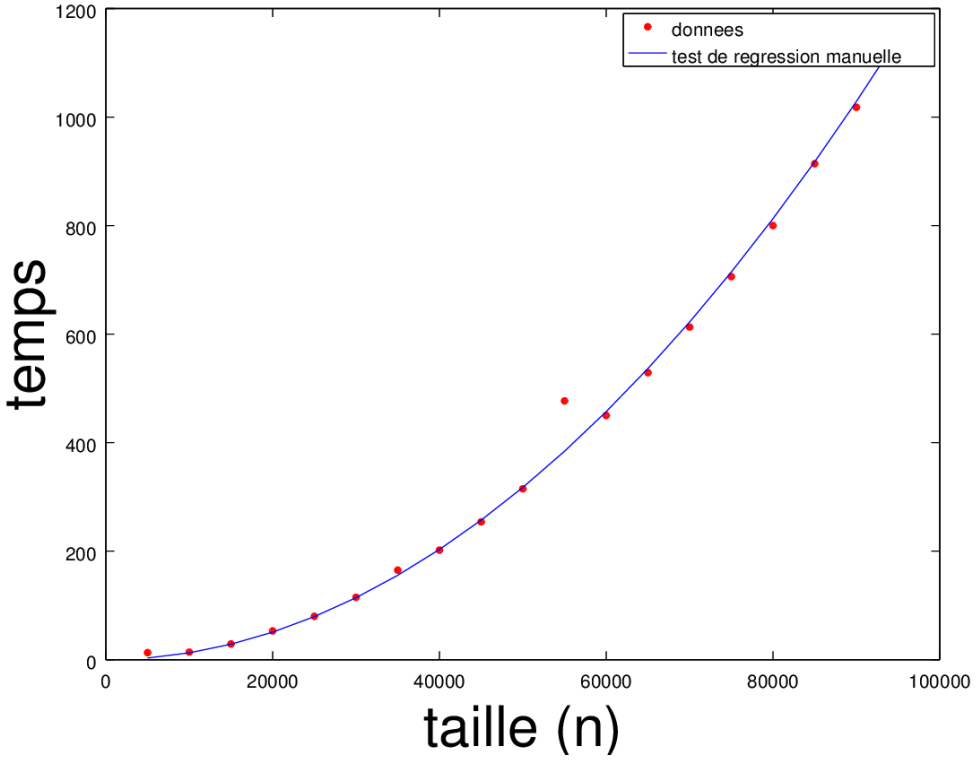
Pour le tri insertion, pour trier 60 000 000 de résultats il faut :
— 33 ms dans le meilleur cas (croissant)
— 380 minutes dans un cas normal
— 12 jours dans le pire des cas
Avec les courbes correspondantes : On obtient une complexité en $O(n^2)$
Pour le tri fusion, on obtient un temps $<$ 10 secondes : L'analyse de l'algorithme de permutation



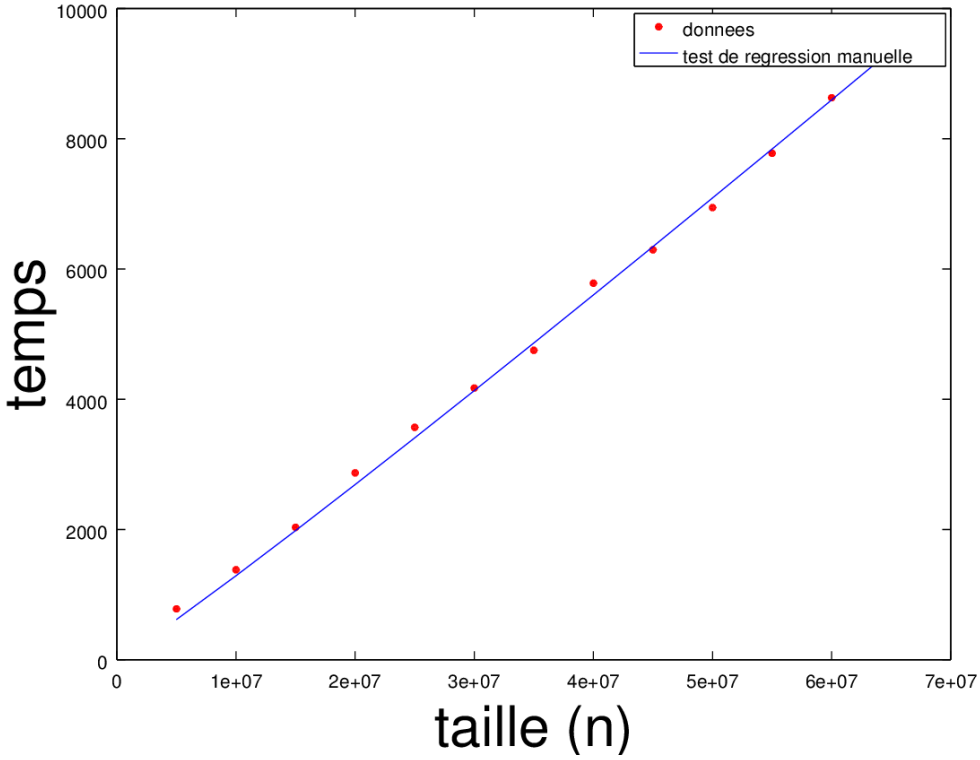
Tri insertion cas croissant



Tri insertion cas aléatoire



Tri insertion cas décroissant


donnees
test de regression manuelle
temps
taille (n)
0
2000
4000
6000
8000
10000
0
1e+07
2e+07
3e+07
4e+07
5e+07
6e+07
7e+07
Tri fusion

(qui cherche toutes les combinaisons avec les n premières lettres de l'alphabet) de la partie 3 nous a donné une complexité factorielle. On remarque qu'elle est très difficile à tracer et que son temps est très vite insupportable.

4 Conclusion

Dans ce TP, nous avons donc pu découvrir et implémenter 3 algorithmes de complexité différentes. Nous pouvons en conclure que $O(n!) >>>> O(n^2) >>>> O(nlogn) > O(n)$