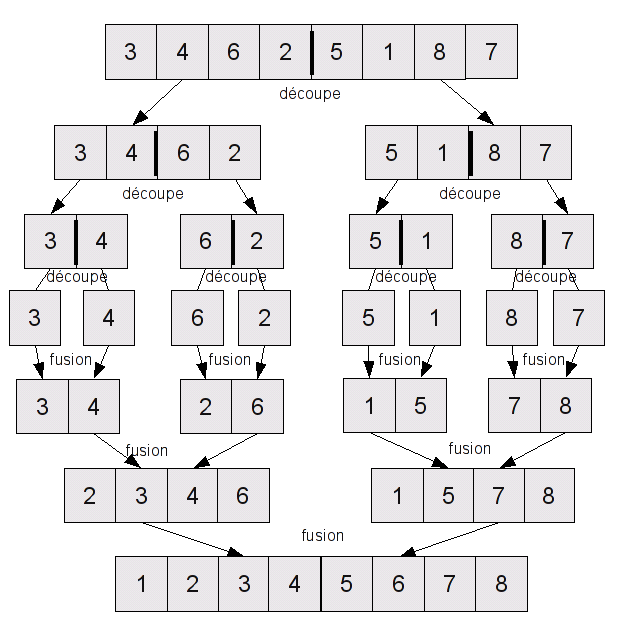
Pseudo code tri par fusion

Le pseudo-code de cet algorithme de tri à été copié du livre recommendu pour le cours ÉLÉ440 s’intitulant : «Introduction à l’arithmitique » à la page 27. Le tri fusion à comme fonctionnement de couper le problème en deux jusqu’à ce qu’il ne reste plus que des nombre isolés. Par la suite il compare deux par deux des blocs de nombre puis les mets en ordre pour former à la fin un seul bloc trié.



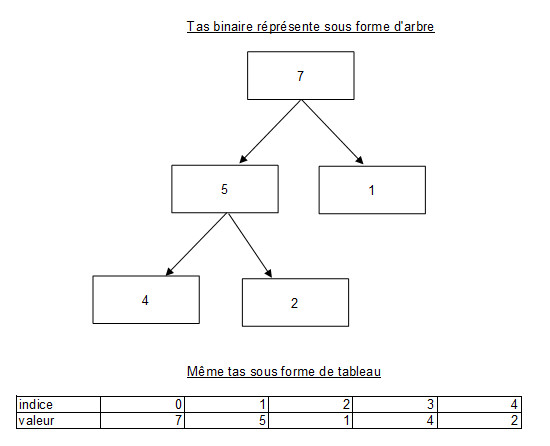
https://openclassrooms.com/courses/algorithmique-pour-l-apprenti-programmeur/introduction-au-probleme-du-tri

Pseudo code tri pigeonnier

Le pseudo-code de cet algorithme à été créé à la main depuis les informations données dans le cours de ÉLÉ440. Le principe étant de créer une case mémoire pour chacune des valeurs possibles dans le tableau reçu puis compter combien de fois on peut retrouver chaque valeurs dans le tableau.

Pseudo code tri par tas

Le pseudo-code de cet algorithme à été créé à la main depuis les informations données dans le cours d’ÉLÉ440. Le principe de cet algorithme est de créer un arbre en divisant toujours les valeurs en 2 sous-valeurs créant des nœuds et des feuilles la premiere donnee du tableau à la pointe de l’arbre. Tous les nœuds doivent avoir une valeur plus grande que leurs sous-nœuds ou feuilles. En appliquant cette loi à tous les nœud en partant de la pointe, échanger les valeurs des nœuds avec ceux d’en dessous puis à chaque fois qu’un nombre est échangé, vérifier que le chiffre échangé est toujours plus bas que le nœud du haut. Une fois terminé, la valeur de la pointe devrait être la valeur la plus grande du tableau. Il faut maintenant permuter la première valeur du tableau et la dernière pour envoyer un très petit chiffre à la pointe. Le but étant de sortir un tableau en ordre croissant, la plus grande valeur est maintenant à la fin du tableau. Il est important de faire abstraction des valeurs placé à la fin dans le reste de l’algorithme pour pas faire remonter des valeurs déjà triés. Puis on répète le processus d’échange du dessus de la piramide autant de fois qu’il y a de données.



https://openclassrooms.com/courses/le-tri-par-tas

Tri par tas

Analyse théorique

L’analyse asymptotique du tri par tas donne une formule plustot grande ce qui sous-entend une efficacité moindre. N2\*(int(racine(N))) décrit le plus grand nombre de fois que l’algorithme passera à un seul endroit. On peut remarquer avec la table des instructions minimum et maximum qu’il est possible de sauter des instructions grâce aux conditions. Cependent, aucune boucle se trouve à l’intérieur d’une condition donc il doit quand même passer à travers les trois boucles même si les données sont déjà tous triés. En ne regardant que les boucles, on remarque que les minimums et maximums sont pareil ce qui n’est pas efficace. Pour que ce soit rapide, l’algorithme doit être capable de détecter rapidement si le tableau est déjà trié.

|  |  |  |
| --- | --- | --- |
| ligne | Min | Max |
| 1 | N | N |
| 2 | N\*N | N\*N |
| 3 | N\*N | N\*N |
| 4 | 0 | N\*N |
| 5 | 0 | N\*N |
| 6 | 0 | N\*N |
| 7 | 0 | N\*N |
| 9 | N\*N | N\*N |
| 10 | 0 | N\*N |
| 11 | 0 | N\*N |
| 14 | N\*N+1 | N\*N\*(int(racine(N)+1)) |
| 15 | 0 | N\*N\*(int(racine(N))) |
| 16 | 0 | N\*N\*(int(racine(N))) |

Tri pas pigeonnier

On peut remarquer rapidement que la formule asymptotique minimal et maximal pour le tri par pigeonnier sont pareil. Ceci signifie que peu importe le degré de désordre, il y aura toujours le même nombre d’instructions utilisées. Aucune condition altère le cours de cette algorithme. C’est un algorithme non-efficace puisqu’il ne s’adapte pas aux données qui lui sont imposés et donc prend toujours le temps maximal pour trier un tableau. Cependant, la formule asymptotique est R2+NR (R étant le rang des données du tableau), donc plus petit le rang, plus rapide l’algorithme. Bref, le tri par pigeonnier est un mauvais algorithme de tri mais peut être plus que d’autres lorsque le rang est petit.

|  |  |  |
| --- | --- | --- |
| Ligne | Min | Max |
| 1 | R | R |
| 2 | R | R |
| 3 | N | N |
| 4 | N | N |
| 5 | R | R |
| 6 | R(N+R) | R(N+R) |
| 7 | R\*N | R\*N |

Tri par fusion

|  |  |  |
| --- | --- | --- |
| Ligne | Min | Max |
| 4 | N/2 | (N/2) |
| 5 | N/2 | (N/2) |
| 6 | N/2 | N/2 |
| 7 | N/2 | N/2 |
| 12 | N | N |
| 13 | N | N |
| 14 | 0 | N |
| 15 | 0 | N |
| 16 | 0 | N |
| 17 | 0 | N |

Tri par base

tri par pigeonnier

On peut remarquer ici que le nombre de données ne fait qu’augmenter linéairement le nombre d’instructions et donc de temps.

On remarque avec ce graphique que quand le rang augmente, le nombre d’instructions augmente rapidement à chaque incrémentation.

On remarque que peu importe le degré de désordre, l’algorithme garde le même nombre d’instruction dépendamment du nombre de données à traiter.

Tri par tas

On peut constater que la performance de cet algorithme détériore rapidement avec des grand nombre de données à trier.

On voit sur le graphique ci-haut que le rang n’affecte pas la performance de cet algorithme.

Le graphique ci-haut illustre bien que le degré de désordre n’a pas d’effet sur l’algorithme.

Pseudo code du tri par base

Pour toutes les puissances de 10

initialiser le tableaux de comptage à zéro.

Pour tout les nombres à trier

Trouver le chiffre de la puissance de 10 correspondante

Incrémenter de 1 la case correspondant du tableau de compteurs

Pour toutes les cases du tableau de compteurs

additionner la valeur de la case (i) avec celle de la case adjacente (i+1))

Pour toutes les donnees à trier,

Trouver le chiffre de la puissance de 10 correspondante

diminuer la valeur de la case du tableau de compteurs de 1

Placer la donnée complète dans la nouvelle matrice à la case précisé par le nombre de la case du tableau de compteur.

Analyse theorique - Tri par base

D’après l’analyse théorique, le tri par base utilise le même nombre d’instructions peu importe le désordre et le rang des données qui lui sont envoyé. Il est cependant inportant de mentionner que sa formule asymptotique est R\*N ce qui est un excellent résultat pour de grand nombre de données et le rang n’est pas nécessairement ce qui ralentira l’algorithme

|  |  |  |
| --- | --- | --- |
| ligne | Min | Max |
| 1 | R | R |
| 2 | R | R |
| 3 | R | R |
| 4 | R\*N | R\*N |
| 6 | R | R |
| 7 | R\*10 | R\*10 |
| 8 | R | R |
| 9 | R\*N | R\*N |
|  |  |  |
|  | R = constant |  |