Le tri postal

avec des seaux

Jean-Marc.Vincent@univ-grenoble-alpes.fr[1]

[1] Laboratoire LIG
Équipe-Projet INRIA POLARIS
Université Grenoble-Alpes

Algorithmique et Modélisation
L3-INFO

I. **LE PROBLÈME : Trier le courrier**

II. **ALGORITHME : Bucket sort**

III. **EXEMPLES**

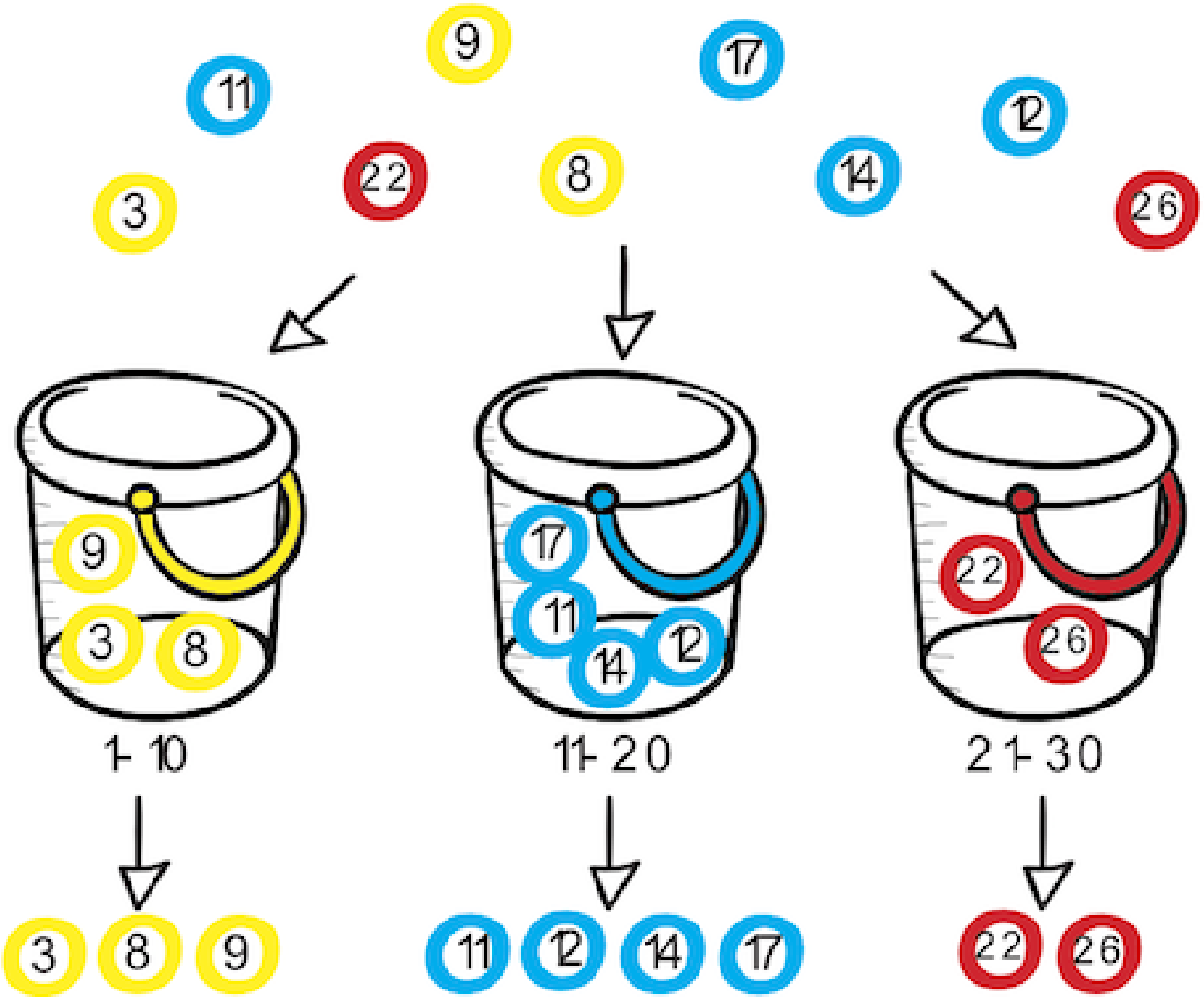
PRINCIPE DU TRI POSTAL

Domaine public, wikipedia

PRINCIPE DU TRI POSTAL

Extrait de sud-ouest: Pau : pourquoi ça a coincé au centre de tri postal

PRINCIPE



Université d'Édimbourg

MISE EN ŒUVRE EN INFORMATIQUE SANS ORDINATEUR

Université d'Édimbourg

TRI PAR PAQUETS

Extrait de *Introduction à l'algorithmique*.

L'idée sous-jacente au tri par paquets est la suivante : on divise l'intervalle $[0, 1)$ en n sous-intervalles de même taille, ou ***paquets***, puis on distribue les n nombres de l'entrée dans les différents paquets. Comme les entrées sont distribuées de manière uniforme sur $[0, 1)$, on n'escompte pas qu'un paquet contienne beaucoup de nombres. Pour produire le résultat, on se contente de trier les nombres de chaque paquet, puis de parcourir tous les paquets, dans l'ordre, en énumérant les éléments de chacun.

Notre code du tri par paquets suppose que l'entrée est un tableau à n éléments A et que chaque élément $A[i]$ du tableau satisfait à $0 \leqslant A[i] < 1$. Le code exige un tableau auxiliaire $B[0\,..\,n-1]$ de listes chaînées (paquets) et suppose qu'il existe un mécanisme pour la gestion de ce genre de listes. (La section 10.2 explique comment implémenter les opérations basiques de liste chaînée.)

```
TRI-PAQUETS(A)
1  n ← longueur[A]
2  pour i ← 1 à n
3      faire insérer A[i] dans liste B[⌊nA[i]⌋]
4  pour i ← 0 à n − 1
5      faire trier liste B[i] via tri par insertion
6  concaténer les listes B[0], B[1], ..., B[n − 1] dans l'ordre
```

La figure 8.4 illustre le fonctionnement du tri par paquets sur un tableau de 10 nombres.

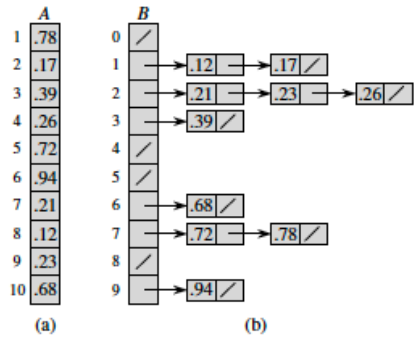


Figure 8.4 Fonctionnement de TRI-PAQUETS. **(a)** Le tableau en entrée $A[1\,..\,10]$. **(b)** Le tableau $B[0\,..\,9]$ de listes (paquets) triées, après la ligne 5 de l'algorithme. Le paquet i contient des valeurs appartenant à l'intervalle semi-ouvert $[i/10, (i+1)/10)$. Le tableau trié consiste en une concaténation ordonnée des listes $B[0], B[1], \ldots, B[9]$.

ligne 3 : $\lfloor x \rfloor$ est la partie entière inférieure de x, (le plus grand entier n inférieur ou égal à x)

On souhaite trier des nombres réels et on fait l'hypothèse que ces nombres ont été tirés aléatoirement sur l'intervalle $[0, 1[$ selon une loi uniforme (les tirages sont supposés indépendants).

On retrouve dans cet algorithme la même idée que pour les tables de hachage. les éléments sont placés dans les cases du tableau en fonction de leur valeur. De plus, dans ce cas, le nombre de places est égal au nombre d'objets à placer. Dans un monde idéal, il n'y aurait qu'un seul objet par case, le hasard va faire qu'une case pourra en contenir plusieurs, mais pas beaucoup.

UGA Université Grenoble Alpes

TRI PAR PAQUETS (1)

Extrait de *Introduction à l'algorithmique*.

L'idée sous-jacente au tri par paquets est la suivante : on divise l'intervalle $[0, 1)$ en n sous-intervalles de même taille, ou ***paquets***, puis on distribue les n nombres de l'entrée dans les différents paquets. Comme les entrées sont distribuées de manière uniforme sur $[0, 1)$, on n'escompte pas qu'un paquet contienne beaucoup de nombres. Pour produire le résultat, on se contente de trier les nombres de chaque paquet, puis de parcourir tous les paquets, dans l'ordre, en énumérant les éléments de chacun.

Notre code du tri par paquets suppose que l'entrée est un tableau à n éléments A et que chaque élément $A[i]$ du tableau satisfait à $0 \leqslant A[i] < 1$. Le code exige un tableau auxiliaire $B[0..n-1]$ de listes chaînées (paquets) et suppose qu'il existe un mécanisme pour la gestion de ce genre de listes. (La section 10.2 explique comment implémenter les opérations basiques de liste chaînée.)

TRI-PAQUETS(A)
1 $n \leftarrow longueur[A]$
2 **pour** $i \leftarrow 1$ à n
3 **faire** insérer $A[i]$ dans liste $B[\lfloor nA[i] \rfloor]$
4 **pour** $i \leftarrow 0$ à $n-1$
5 **faire** trier liste $B[i]$ *via* tri par insertion
6 concaténer les listes $B[0], B[1], \ldots, B[n-1]$ dans l'ordre

La figure 8.4 illustre le fonctionnement du tri par paquets sur un tableau de 10 nombres.

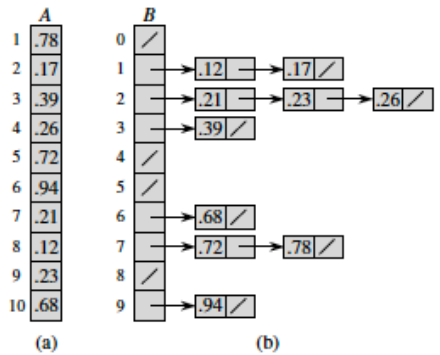


Figure 8.4 Fonctionnement de TRI-PAQUETS. **(a)** Le tableau en entrée $A[1..10]$. **(b)** Le tableau $B[0..9]$ de listes (paquets) triées, après la ligne 5 de l'algorithme. Le paquet i contient des valeurs appartenant à l'intervalle semi-ouvert $[i/10, (i+1)/10)$. Le tableau trié consiste en une concaténation ordonnée des listes $B[0], B[1], \ldots, B[9]$.

ligne 3 : $\lfloor x \rfloor$ est la partie entière inférieure de x. (le plus

Quelle est la loi de N_i ?

A. Bernoulli
B. Binomiale
C. Poisson
D. Géométrique

UGA Université Grenoble Alpes

TRI PAR PAQUETS (1)

Extrait de *Introduction à l'algorithmique*.

L'idée sous-jacente au tri par paquets est la suivante : on divise l'intervalle $[0, 1)$ en n sous-intervalles de même taille, ou ***paquets***, puis on distribue les n nombres de l'entrée dans les différents paquets. Comme les entrées sont distribuées de manière uniforme sur $[0, 1)$, on n'escompte pas qu'un paquet contienne beaucoup de nombres. Pour produire le résultat, on se contente de trier les nombres de chaque paquet, puis de parcourir tous les paquets, dans l'ordre, en énumérant les éléments de chacun.

Notre code du tri par paquets suppose que l'entrée est un tableau à n éléments A et que chaque élément $A[i]$ du tableau satisfait à $0 \leqslant A[i] < 1$. Le code exige un tableau auxiliaire $B[0\,..\,n-1]$ de listes chaînées (paquets) et suppose qu'il existe un mécanisme pour la gestion de ce genre de listes. (La section 10.2 explique comment implémenter les opérations basiques de liste chaînée.)

TRI-PAQUETS(A)
1 $n \leftarrow$ *longueur*$[A]$
2 **pour** $i \leftarrow 1$ à n
3 **faire** insérer $A[i]$ dans liste $B[\lfloor nA[i] \rfloor]$
4 **pour** $i \leftarrow 0$ à $n-1$
5 **faire** trier liste $B[i]$ *via* tri par insertion
6 concaténer les listes $B[0], B[1], \ldots, B[n-1]$ dans l'ordre

La figure 8.4 illustre le fonctionnement du tri par paquets sur un tableau de 10 nombres.

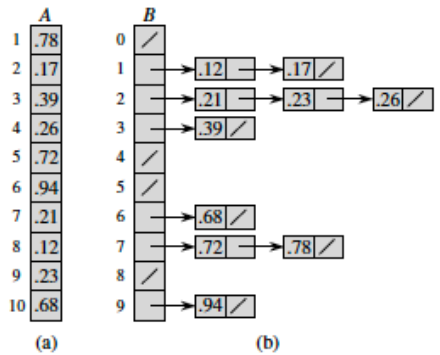


Figure 8.4 Fonctionnement de TRI-PAQUETS. **(a)** Le tableau en entrée $A[1\,..\,10]$. **(b)** Le tableau $B[0\,..\,9]$ de listes (paquets) triées, après la ligne 5 de l'algorithme. Le paquet i contient des valeurs appartenant à l'intervalle semi-ouvert $[i/10, (i+1)/10)$. Le tableau trié consiste en une concaténation ordonnée des listes $B[0], B[1], \ldots, B[9]$.

ligne 3 : $\lfloor x \rfloor$ est la partie entière inférieure de x. (le plus

Quelle est la loi de N_i ?

Comme les tirages sont aléatoires et uniformes sur $[0, 1[$ la probabilité qu'un nombre soit placé dans la case i est la probabilité que la valeur de ce nombre soit dans $[\frac{i}{n}, \frac{i+1}{n}[$, c'est à dire $\frac{1}{n}$.
On a donc une séquence de tirages indépendants suivant la loi de Bernoulli $\mathcal{B}(\frac{1}{n})$, donc

$$\mathbb{P}(N_i = k) = \binom{n}{k} \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k}$$

et on reconnaît une loi binomiale $\mathcal{B}in(n, p)$ avec $p = \frac{1}{n}$

Calculer la moyenne de N_i.

D'après le cours la moyenne est $n \times p = n\frac{1}{n} = 1$ ce qui est logique !

Université Grenoble Alpes

TRI PAR PAQUETS (2)

Extrait de *Introduction à l'algorithmique*.

L'idée sous-jacente au tri par paquets est la suivante : on divise l'intervalle $[0, 1)$ en n sous-intervalles de même taille, ou ***paquets***, puis on distribue les n nombres de l'entrée dans les différents paquets. Comme les entrées sont distribuées de manière uniforme sur $[0, 1)$, on n'escompte pas qu'un paquet contienne beaucoup de nombres. Pour produire le résultat, on se contente de trier les nombres de chaque paquet, puis de parcourir tous les paquets, dans l'ordre, en énumérant les éléments de chacun.

Notre code du tri par paquets suppose que l'entrée est un tableau à n éléments A et que chaque élément $A[i]$ du tableau satisfait à $0 \leqslant A[i] < 1$. Le code exige un tableau auxiliaire $B[0..n-1]$ de listes chaînées (paquets) et suppose qu'il existe un mécanisme pour la gestion de ce genre de listes. (La section 10.2 explique comment implémenter les opérations basiques de liste chaînée.)

TRI-PAQUETS(A)
1 $n \leftarrow longueur[A]$
2 **pour** $i \leftarrow 1$ à n
3 **faire** insérer $A[i]$ dans liste $B[\lfloor nA[i] \rfloor]$
4 **pour** $i \leftarrow 0$ à $n-1$
5 **faire** trier liste $B[i]$ *via* tri par insertion
6 concaténer les listes $B[0], B[1], \ldots, B[n-1]$ dans l'ordre

La figure 8.4 illustre le fonctionnement du tri par paquets sur un tableau de 10 nombres.

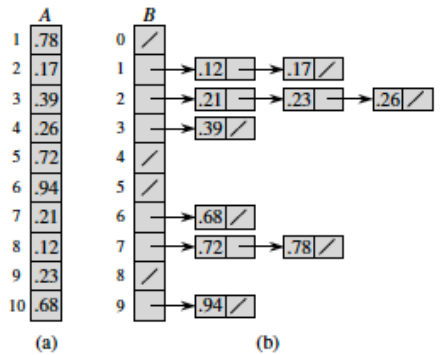


Figure 8.4 Fonctionnement de TRI-PAQUETS. **(a)** Le tableau en entrée $A[1..10]$. **(b)** Le tableau $B[0..9]$ de listes (paquets) triées, après la ligne 5 de l'algorithme. Le paquet i contient des valeurs appartenant à l'intervalle semi-ouvert $[i/10, (i+1)/10)$. Le tableau trié consiste en une concaténation ordonnée des listes $B[0], B[1], \ldots, B[9]$.

ligne 3 : $\lfloor x \rfloor$ est la partie entière inférieure de x. (le plus

Quel est le coût au pire du tri utilisé ligne 5.

A. $\mathcal{O}(1)$
B. $\mathcal{O}(N_i)$
C. $\mathcal{O}(N_i^2)$
D. $\mathcal{O}(N^2)$

TRI PAR PAQUETS (2)

Extrait de *Introduction à l'algorithmique*.

L'idée sous-jacente au tri par paquets est la suivante : on divise l'intervalle $[0, 1)$ en n sous-intervalles de même taille, ou ***paquets***, puis on distribue les n nombres de l'entrée dans les différents paquets. Comme les entrées sont distribuées de manière uniforme sur $[0, 1)$, on n'escompte pas qu'un paquet contienne beaucoup de nombres. Pour produire le résultat, on se contente de trier les nombres de chaque paquet, puis de parcourir tous les paquets, dans l'ordre, en énumérant les éléments de chacun.

Notre code du tri par paquets suppose que l'entrée est un tableau à n éléments A et que chaque élément $A[i]$ du tableau satisfait à $0 \leqslant A[i] < 1$. Le code exige un tableau auxiliaire $B[0..n-1]$ de listes chaînées (paquets) et suppose qu'il existe un mécanisme pour la gestion de ce genre de listes. (La section 10.2 explique comment implémenter les opérations basiques de liste chaînée.)

```
TRI-PAQUETS(A)
1  n ← longueur[A]
2  pour i ← 1 à n
3      faire insérer A[i] dans liste B[⌊nA[i]⌋]
4  pour i ← 0 à n − 1
5      faire trier liste B[i] via tri par insertion
6  concaténer les listes B[0], B[1], ..., B[n − 1] dans l'ordre
```

La figure 8.4 illustre le fonctionnement du tri par paquets sur un tableau de 10 nombres.

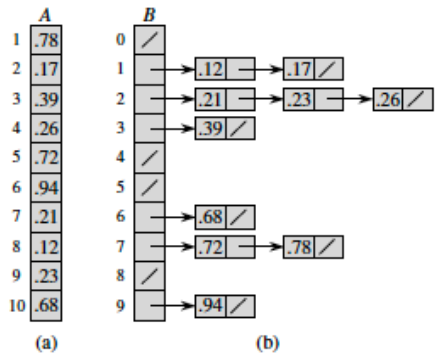


Figure 8.4 Fonctionnement de TRI-PAQUETS. **(a)** Le tableau en entrée $A[1..10]$. **(b)** Le tableau $B[0..9]$ de listes (paquets) triées, après la ligne 5 de l'algorithme. Le paquet i contient des valeurs appartenant à l'intervalle semi-ouvert $[i/10, (i+1)/10)$. Le tableau trié consiste en une concaténation ordonnée des listes $B[0], B[1], \ldots, B[9]$.

ligne 3 : $\lfloor x \rfloor$ est la partie entière inférieure de x. (le plus

Quel est le coût au pire du tri utilisé ligne 5.

On a un tri par insertion ligne 5, on a vu en cours et en TD que le coût d'une insertion du kieme dans un tableau trié était majoré par $k-1$ (on décale tous les éléments vers la droite), donc en faisant l'itération pour tous les éléments on obtient un coût de

$$1+2+3+\cdots+(N_i-1) = \frac{N_i(N_i-1)}{2} \leqslant N_i^2$$

TRI PAR PAQUETS (3)

Extrait de *Introduction à l'algorithmique*.

L'idée sous-jacente au tri par paquets est la suivante : on divise l'intervalle $[0, 1)$ en n sous-intervalles de même taille, ou ***paquets***, puis on distribue les n nombres de l'entrée dans les différents paquets. Comme les entrées sont distribuées de manière uniforme sur $[0, 1)$, on n'escompte pas qu'un paquet contienne beaucoup de nombres. Pour produire le résultat, on se contente de trier les nombres de chaque paquet, puis de parcourir tous les paquets, dans l'ordre, en énumérant les éléments de chacun.

Notre code du tri par paquets suppose que l'entrée est un tableau à n éléments A et que chaque élément $A[i]$ du tableau satisfait à $0 \leqslant A[i] < 1$. Le code exige un tableau auxiliaire $B[0\,..\,n-1]$ de listes chaînées (paquets) et suppose qu'il existe un mécanisme pour la gestion de ce genre de listes. (La section 10.2 explique comment implémenter les opérations basiques de liste chaînée.)

TRI-PAQUETS(A)
1 $n \leftarrow longueur[A]$
2 **pour** $i \leftarrow 1$ à n
3 **faire** insérer $A[i]$ dans liste $B[\lfloor nA[i] \rfloor]$
4 **pour** $i \leftarrow 0$ à $n-1$
5 **faire** trier liste $B[i]$ *via* tri par insertion
6 concaténer les listes $B[0], B[1], \ldots, B[n-1]$ dans l'ordre

La figure 8.4 illustre le fonctionnement du tri par paquets sur un tableau de 10 nombres.

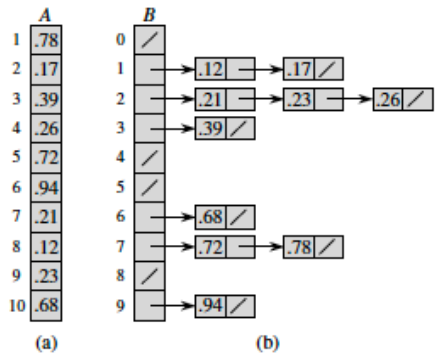


Figure 8.4 Fonctionnement de TRI-PAQUETS. **(a)** Le tableau en entrée $A[1\,..\,10]$. **(b)** Le tableau $B[0\,..\,9]$ de listes (paquets) triées, après la ligne 5 de l'algorithme. Le paquet i contient des valeurs appartenant à l'intervalle semi-ouvert $[i/10, (i+1)/10)$. Le tableau trié consiste en une concaténation ordonnée des listes $B[0], B[1], \ldots, B[9]$.

ligne 3 : $\lfloor x \rfloor$ est la partie entière inférieure de x. (le plus

Donner un majorant de la moyenne de N_i^2.

TRI PAR PAQUETS (3)

Extrait de *Introduction à l'algorithmique*.

L'idée sous-jacente au tri par paquets est la suivante : on divise l'intervalle $[0, 1)$ en n sous-intervalles de même taille, ou ***paquets***, puis on distribue les n nombres de l'entrée dans les différents paquets. Comme les entrées sont distribuées de manière uniforme sur $[0, 1)$, on n'escompte pas qu'un paquet contienne beaucoup de nombres. Pour produire le résultat, on se contente de trier les nombres de chaque paquet, puis de parcourir tous les paquets, dans l'ordre, en énumérant les éléments de chacun.

Notre code du tri par paquets suppose que l'entrée est un tableau à n éléments A et que chaque élément $A[i]$ du tableau satisfait à $0 \leqslant A[i] < 1$. Le code exige un tableau auxiliaire $B[0..n-1]$ de listes chaînées (paquets) et suppose qu'il existe un mécanisme pour la gestion de ce genre de listes. (La section 10.2 explique comment implémenter les opérations basiques de liste chaînée.)

TRI-PAQUETS(A)
1 $n \leftarrow$ *longueur*$[A]$
2 **pour** $i \leftarrow 1$ à n
3 **faire** insérer $A[i]$ dans liste $B[\lfloor nA[i] \rfloor]$
4 **pour** $i \leftarrow 0$ à $n-1$
5 **faire** trier liste $B[i]$ *via* tri par insertion
6 concaténer les listes $B[0], B[1], \ldots, B[n-1]$ dans l'ordre

La figure 8.4 illustre le fonctionnement du tri par paquets sur un tableau de 10 nombres.

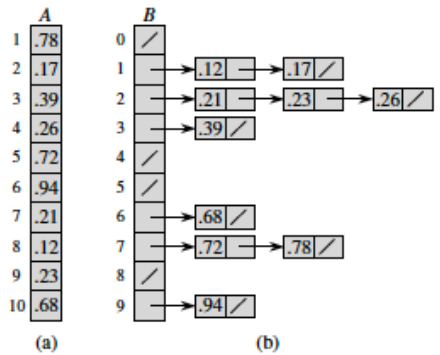


Figure 8.4 Fonctionnement de TRI-PAQUETS. **(a)** Le tableau en entrée $A[1..10]$. **(b)** Le tableau $B[0..9]$ de listes (paquets) triées, après la ligne 5 de l'algorithme. Le paquet i contient des valeurs appartenant à l'intervalle semi-ouvert $[i/10, (i+1)/10)$. Le tableau trié consiste en une concaténation ordonnée des listes $B[0], B[1], \ldots, B[9]$.

ligne 3 : $\lfloor x \rfloor$ est la partie entière inférieure de x. (le plus

Donner un majorant de la moyenne de N_i^2.

On veut majorer $\mathbb{E}(N_i^2)$. Pour calculer sa valeur on remarque que la variance d'une loi binomiale est $\mathbb{V}arN_i = np(1-p) = 1 - \frac{1}{n}$. Or, en utilisant l'indication $\mathbb{V}arN_i = \mathbb{E}(N_i^2) - (\mathbb{E}N_i)^2$ et le fait que $\mathbb{E}N_i = 1$

$$\mathbb{E}(N_i^2) = 2 - \frac{1}{n} \leqslant 2$$

Donc le coût moyen de tri par case est majoré par 2.

TRI PAR PAQUETS (4)

Extrait de *Introduction à l'algorithmique*.

L'idée sous-jacente au tri par paquets est la suivante : on divise l'intervalle $[0, 1)$ en n sous-intervalles de même taille, ou ***paquets***, puis on distribue les n nombres de l'entrée dans les différents paquets. Comme les entrées sont distribuées de manière uniforme sur $[0, 1)$, on n'escompte pas qu'un paquet contienne beaucoup de nombres. Pour produire le résultat, on se contente de trier les nombres de chaque paquet, puis de parcourir tous les paquets, dans l'ordre, en énumérant les éléments de chacun.

Notre code du tri par paquets suppose que l'entrée est un tableau à n éléments A et que chaque élément $A[i]$ du tableau satisfait à $0 \leqslant A[i] < 1$. Le code exige un tableau auxiliaire $B[0\,..\,n-1]$ de listes chaînées (paquets) et suppose qu'il existe un mécanisme pour la gestion de ce genre de listes. (La section 10.2 explique comment implémenter les opérations basiques de liste chaînée.)

TRI-PAQUETS(A)
1 $n \leftarrow$ *longueur*$[A]$
2 **pour** $i \leftarrow 1$ **à** n
3 **faire** insérer $A[i]$ dans liste $B[\lfloor nA[i] \rfloor]$
4 **pour** $i \leftarrow 0$ **à** $n-1$
5 **faire** trier liste $B[i]$ *via* tri par insertion
6 concaténer les listes $B[0], B[1], \ldots, B[n-1]$ dans l'ordre

La figure 8.4 illustre le fonctionnement du tri par paquets sur un tableau de 10 nombres.

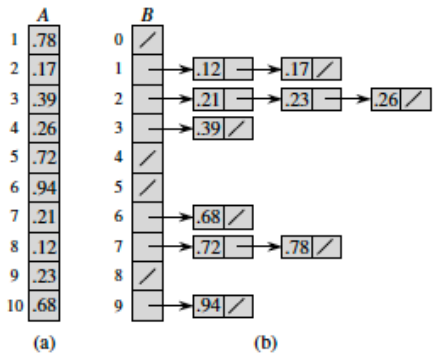


Figure 8.4 Fonctionnement de TRI-PAQUETS. **(a)** Le tableau en entrée $A[1\,..\,10]$. **(b)** Le tableau $B[0\,..\,9]$ de listes (paquets) triées, après la ligne 5 de l'algorithme. Le paquet i contient des valeurs appartenant à l'intervalle semi-ouvert $[i/10, (i+1)/10)$. Le tableau trié consiste en une concaténation ordonnée des listes $B[0], B[1], \ldots, B[9]$.

ligne 3 : $\lfloor x \rfloor$ est la partie entière inférieure de x. (le plus

En déduire la complexité moyenne de votre algorithme, commenter ce résultat.

Le coût moyen de cet algorithme est donc en $\mathcal{O}(n)$ (tri en temps linéaire), ce qui peut paraître surprenant car nous avons vu que le problème du tri était en $\mathcal{O}(n \log n)$. En fait ici on ne fait des comparaisons que dans le tri des cases, d'autres comparaisons sont cachées dans le calcul du numéro de la case (il faut $\log n$ bits pour identifier la case, tous les bits sont significatifs, donc on retrouve le $n \log n$. De plus on a une information complémentaire sur les données ce qui est une restriction au problème général.

UGA Université Grenoble Alpes

PROTOTYPAGE

Code python

```
def bucket_sort(L):
    """ tri par bucket (seau)
    parametres :
      L : liste de nombres dans [0,1[
    resultat :
      liste des elements de L tries
    methode :
      construction d'une liste de paquets tries, adressage de la liste par la
      valeur de l'element a trier

    """
    n = len(L)
    # distribution dans les buckets
    bucket_list = [[] for i in range(n)]
    for e in L:
        insert_bucket(e,bucket_list[math.floor(e*n)])
    # concatenation des listes resultat
    resultat =[]
    for l in bucket_list:
        resultat = resultat + l
    return resultat
```