Tris bulles

Christian Jean CYRILLE

12 janvier 2020

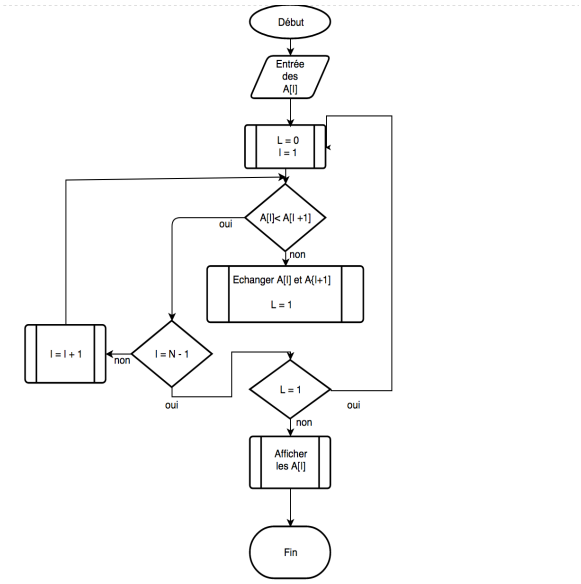
1 Principe

On compare successivement 2 éléments consécutifs en partant du premier élément :

- s'ils sont dans le bon ordre, on passe aux suivants
- sinon on les échange et on passe aux suivants
- Après un examen de tous les couples, l'élément le plus grand se retrouve en dernière position.
- On recommence ensuite pour les $n-1$ derniers éléments afin d'amener le plus grand d'entre eux en avant-dernière position.
- Après au plus $n-1$ examens, le tableau est trié.

On peut aussi ramener les éléments dans l'ordre croissant en cherchant chaque fois le plus petit. On appelle ce tri du nom de tri bulle (en anglais bubble sort) car les plus petits éléments remontent petit à petit à leur place comme des bulles d'air à la surface de l'eau.

Le tri bulles croissant est illustré par l'ordinogramme suivant :
On utilise une variable drapeau("flag") qui vaut 0 lorsqu'il n'y a pas eu d'échanges sinon elle vaut 1.



2 Algorithmes

2.1 Algorithme dans l'ordre croissant

Le processus peut être programmé très facilement avec deux boucles : la première boucle assure la lecture de tous les éléments du début jusqu'à la fin ; la seconde permet le déplacement du plus grand élément à la fin de la liste, celle-ci étant à chaque fois privée de son dernier élément grâce à la progression de l'indice de la première boucle.

```
Procedure BUBBLE_SORT(var A : array[1..N] of integer;
(* version lourde *)
var I,J : integer;
begin
       for I := 2 to N
        do begin
                  for J := N downto I
                       do if A[J - 1] > A[J] then ECHANGE(A[J-1],A[J])
               end;
 end;
```

Mais il se peut qu'on n'ait pas besoin de $N-1$ parcours à reculons.

1. Dès qu'il n'y a plus d'échanges, le tableau est en fait déjà trié.
2. De plus, si on note K l'indice du dernier $A[J]$ qui a été échangé , la fois suivante, au lieu de reculer J de N à 2 , il suffit de reculer J de N à $K+1$ car de 1 à K les élément sont tous en place.

```
Procedure BUBBLE_SORT(var A : array[1..N] of integer);
(* version optimisée 1*)
var I,J : integer;
IL_Y_A_EU_ECHANGE : boolean;
begin
       I:=2;
       K :=N;
       repeat
                 IL_Y_A_EU_ECHANGE := false;
                  for J := N downto I
                       do if A[J - 1] > A[J] then
                                                                  begin
                                                                      ECHANGE(A[J-1],A[J]);
                                                                      K := J;
                                                                      IL_Y_A_EU_ECHANGE ::=true;
                                                                  end;
                 I :=K + 1;
          until not(IL_Y_A_EU_ECHANGE ) or (I > N);
 end;
```

Autre variante :

```
Procedure BUBBLE_SORT(var A : array[1..N] of integer);
(* version optimisée 2 *)
var I,J : integer;
IL_Y_A_EU_ECHANGE : boolean;
begin
       I:=2;
       K :=N;
       repeat
                 IL_Y_A_EU_ECHANGE := false;
                  J := N;
                  repeat
                           if A[J - 1] > A[J] then
                                                                  begin
                                                                      ECHANGE(A[J-1],A[J]);
                                                                      K := J;
                                                                      IL_Y_A_EU_ECHANGE ::=true;
                                                                  end;
                            J:= J - 1;
                  until J = I;
           I := I + 1;
          until not(IL_Y_A_EU_ECHANGE ) or (I > N);
 end;
```

3 Complexité

3.1 Tri à bulles classique

On parcourt les n éléments de gauche à droite en échangeant les éléments voisins chaque fois que l'élément à gauche est supérieur à l'élément de droite.
On recommence avec les $n-1$ premiers éléments.
Cela nécessite $\frac{n^2-n}{4}$ décalages et $\frac{n^2-n}{2}$ comparaisons.
Les mauvaise performances de ce tri proviennent du fait que, dans tous les cas, on est sûr d'avoir un nombre de comparaisons équivalent au carré n^2 du nombre n de données à trier. Même sur une liste déjà triée, en effet, l'algorithme se déroulera inexorablement du début jusqu'à la fin.

23	56	89	12	34	45	67	3
23	56	12	34	45	67	3	89
23	12	34	45	56	3	67	89
12	23	34	45	3	56	67	89
12	23	3	34	45	56	67	89
12	3	23	34	45	56	67	89
3	12	23	34	45	56	67	89

1 Quel est le meilleur des cas ?
Déterminer en fonction de n le nombre N_{min} de comparaisons à faire pour trier le tableau T

2 Quel est le pire des cas ?
Déterminer en fonction de n le nombre N_{max} de comparaisons à faire pour trier le tableau T

3.2 Tri à bulles amélioré (Shake)

Tout en reprenant le même principe que le tri bulles classique, le tri bulles optimisé bénéficie de performances améliorées, au prix il est vrai d'un certain allongement du programme. Il s'agit de mémoriser l'emplacement où a été effectuée la dernière permutation pour arrêter l'exploration de la liste : si l'on n'a pas permuté plus loin c'est que cette partie de la liste est déjà ordonnée. En outre, on évite de repartir à chaque fois du début de la liste : le tri est effectué simultanément aux deux extrémités., le plus grand élément remontant en fin de liste, tandis que le plus petit est amené au début.

- Premier passage :
  - → Aller : On parcourt les n éléments de gauche à droite en échangeant les éléments voisins chaque fois que l'élément à gauche est supérieur à l'élément de droite.
  - → Retour : On recommence avec les éléments (à partir d'une marque *rouge* mais dans l'autre sens avec la condition d'échange inverse.

  Deuxième passage :
  - → Aller : On parcourt les éléments de gauche à droite (à partir d'une marque *bleue* en échangeant les éléments voisins chaque fois que l'élément à gauche est supérieur à l'élément de droite.
  - → Retour : On recommence avec les éléments (à partir d'une marque *rouge* mais dans l'autre sens avec la condition d'échange inverse.
- et ainsi de suite ...

Le nombre de comparaisons est forcément plus faible.

Exemple : Table initiale

5	7	3	9	4	1	8	6	12	10	2	13	11

Premier passage :
Aller

5	3	7	4	1	8	6	9	10	2	12	11	13

Retour

1	5	3	7	4	2	8	6	9	10	11	12	13

Deuxième passage :
Aller

1	3	5	4	2	7	6	8	9	10	11	12	13

Retour

1	2	3	5	4	6	7	8	9	10	11	12	13

Troisième passage :
Aller

1	2	3	4	5	6	7	8	9	10	11	12	13

Fin de l'exploration. le drapeau("flag") reste à 0 : le tableau est trié. Il a suffi de 3 passages aller-retour pour trier ce tableau

1 Quel est le meilleur des cas ?
Déterminer en fonction de n le nombre N_{min} de comparaisons à faire pour trier le tableau T

2 Quel est le pire des cas ?
Déterminer en fonction de n le nombre N_{max} de comparaisons à faire pour trier le tableau T

4 Bibilographie

- "Le tri en direct à l'écran" - Article de Philippe MATHIEU - Science et Vie Micro - n° 54 - Octobre 1988
-
-