Prof : FRIKHA Mohamed Heni

Algorithme et programmation
À l'aide de python
« Préparer BAC 2022 »
« Collection de problèmes »

LES ALGORITHMES DE TRI

ET RECHERCHE

Niveau : 4èmes Scientifiques

50 Minutes

3 Nombre premier sûr

Analyse & Algorithme

Implémentation Avec PYTHON

Med Heni Frikha

Un nombre **M** est dit « **nombre premier sûr** », s'il est un nombre premier de la forme **2*p+1** avec p un nombre premier.

Exemples :

Premier → M = 2*P + 1
Premier → P = (M − 1) div 2
Premier
Premier

- ✓ Si **M**= 11, alors **M** est un nombre premier sûr. En effet, **11** est premier et il peut s'écrire sous la forme **2*p+1** où **p = 5** qui est un nombre premier.
- ✓ Si **M= 31**, alors **M** n'est pas un nombre premier sûr. En effet, **31** est premier et il peut s'écrire sous la forme 2*p+1 où **p = 15** qui n'est pas un nombre premier.

N.B. : un nombre entier supérieur à 1 est dit premier s'il n'est divisible que par **1** et par lui-même.

On se propose d'écrire un programme qui permet de :

1. Remplir un tableau **T** par **N** entier strictement supérieurs à 1 (avec **10 ≤ N < 45**).
2. Trier dans l'ordre croissant les éléments premiers sûrs du tableau **T** suivis du reste des éléments sans tri.
3. Afficher le tableau **T** résultat.

Exemple : Pour N = 10 et le tableau T suivant :

T	5	25	59	23	13	47	31	100	7	107
	0	1	2	3	4	5	6	7	8	9

Le programme affichera le contenu du tableau suivant :

T	5	7	23	47	59	107	25	13	31	100
	0	1	2	3	4	5	6	7	8	9

Eléments premiers sûrs triés dans un ordre croissant — Eléments non premiers sûrs

Correction de ....

Med Heni Frikha

```
Fonction PremierSur ( M : entier ): booléen
Debut
    P ← (M-1) div 2
    si Premier(P) = Vrai et Premier(M) = Vrai Alors
        test ← Vrai
    Sinon
        test ← Faux
    Finsi
    Retourner test
Fin
```

T.D.OL ?

```
Fonction Premier ( X : entier ): booléen
Debut
    test ← Vrai                          compteur automatique
    Pour i de 2 à X-1 faire
        si X mod i = 0 Alors
            test ← Faux
        Finsi
    Finpour
    si X < 1 Alors
        test ← Faux
    Finsi
    Retourner test
```

```
    test ← Vrai                          compteur manuel
    i ← 2
    tantque i < X et test = Vrai faire
        si X mod i = 0 Alors
            test ← Faux
        Finsi
        i ← i+1
    Fintantque
    si X < 1 Alors
        test ← Faux
    Finsi
```

Correction de ....

Med Heni Frikha

```
Procedure GrapeTrier (@T:tab, N:entier)
Debut
   {RemplirV par les premiers}
   nV ← 0
   Pour i de 0 à N-1 faire
      Si PremierSur(T[i]) = Vrai Alors
         V[nV] ← T[i]
         nV ← nV+1
      Finsi
   Finpour
   Trier (V, nV)
   Pour i de 0 à N-1 faire
      Si PremierSur(T[i]) = Faux Alors
         V[nV] ← T[i]
         nV ← nV+1
      Finsi
   Finpour

   Pour i de 0 à N-1 faire
      T[i] ← V[i]
   Finpour
```

Correction de ....

Med Heni Frikha

Repeter
test ← Vrai

Pour i de 0 à N-2 faire

Si (PremierSur(T[i]) = Faux et PremierSur(T[i+1]) = Vrai) ou ((PremierSur(T[i]) = Vrai et PremierSur(T[i+1]) = Vrai) et (T[i] > T[i+1])) Alors

x ← T[i]
T[i] ← T[i+1]
T[i+1] ← x
test ← Faux

Finsi

Finpour

jusqu'a test = Vrai

5 Suite... max... min...

50 Minutes — Analyse & Algorithme — Implémentation Avec PYTHON — Med Heni Frikha

Soit un nombre N, un entier naturel de quatre chiffres. On forme à l'aide des quatre chiffres de N, le plus grand entier naturel Max et le plus petit entier naturel min ; leur différence (Max-Min) donne un nombre. On refait le même travail pour ce nouveau nombre et on obtient ainsi une suite. Cette suite est stationnaire c'est-à-dire qu'elle devient constante à partir d'un certain rang.

Ecrire un programme PYTHON qui permet de saisir un entier naturel N formé de quatre chiffres puis calcule et affiche les termes de cette suite jusqu'elle devienne constante

Exemple :

```
Donner un entier de quatre chiffres : 2915
La suite est :
9521 - 1259 = 8262
8622 - 2268 = 6354
6543 - 3456 = 3087
8730 -  378 = 8352
8532 - 2358 = 6174
7641 - 1467 = 6174
```

6 Nouvelle méthode de tri

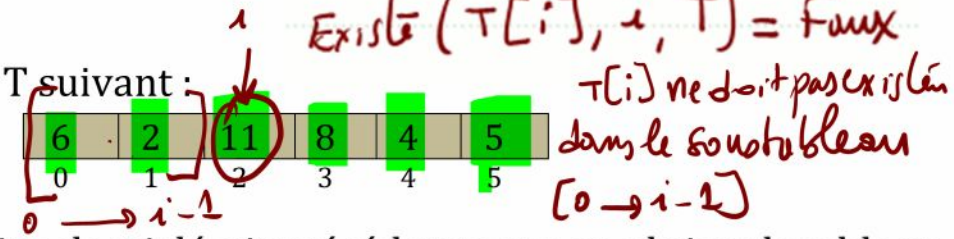
50 Minutes — Analyse & Algorithme — Implémentation Avec PYTHON — Med Heni Frikha

Soit T un tableau contenant des entiers distincts de l'intervalle [0,99].

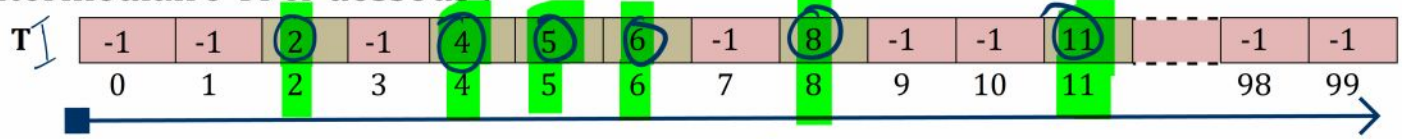
Pour trier dans l'ordre croissant les éléments du tableau T, on propose la méthode suivante :

- Placer chaque élément T[i] dans la case d'indice T[i] d'un tableau intermédiaire TI, sachant que les éléments du tableau TI sont initialisés à -1.
- Placer dans l'ordre tous les entiers différents de -1 du tableau TI, dans le tableau T.

Exemple : pour le tableau T suivant :



Après application du principe de tri décrit précédemment, on obtient le tableau intermédiaire TI ci-dessous :



Correction de ....

Med Heni Frikha

Prog Principal →
- Proc Saisir (N) — $5 \leq N \leq 30$
- Proc Remplir (T, N) — $0 \leq T[i] \leq 99$
- Proc Trier (T, N) { ① Remplir TI ② Remplir de nouveau T
- Proc Afficher (T, N)

① Programme principal

```
Algorithme Sujet6
Debut
    saisir (N)
    Remplir (T,N)
    Trier (T,N)
    Afficher (T,N)
Fin
```

② Declaration

T.D.O.G

objet	Nature/type
N	Entier
T	Tab
Saisir	Procedure
Remplir	Procedure
Trier	Procedure
Afficher	Procedure

T.D.N.T

Nouveau type
Tab = tableau de 30 entiers

③ Algorithmes des sous prog

```
Procedure Saisir (@ N: entier)
Debut
    Repeter
        lire (N)
    jusqu'a 5 ≤ N ≤ 30
Fin
```

Passage par variable/adresse/référence

Correction de ....

Med Heni Frikha

Procedure Afficher (T: tab, N: entier)
Debut
  Pour i de 0 à N-1 faire
    Ecrire (T[i])
  Finpour
Fin

Ecrire (T)

Il ne faut pas afficher directement un tableau

T.D.O.L

objet	Nature/type
i	Entier

Procedure Remplir (@ T: tab, N: entier)
Debut
  Pour i de 0 à N-1 faire
    Repeter
      Lire (T[i])
    jusqu'à 0 ≤ T[i] ≤ 99 et Existe (T[i], i, T) = Faux
  Finpour
Fin

T.D.O.L

objet	Nature/type
i	Entier
Existe	Fonction/booléen

Fonction existe (x: entier, i: entier, T: tab): booléen
Debut
  test ← Faux
  j ← 0
  tantque j < i et test = Faux Faire
    si T[j] = x Alors
      test ← Vrai
    Finsi
    j ← j + 1
  Fintantque
  Retourner Test
Fin

Verifie si x existe ou non dans le sous tableau [0 → i-1]

T.D.O.L

objet	Nature/type
j	Entier
test	booléen

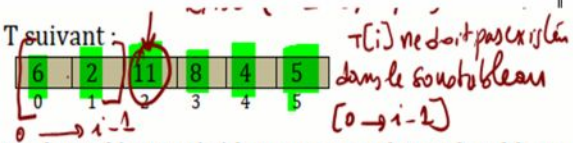
Correction de ....

Med Heni Frikha

```
Procedure Trier (@T:tab, N:entier)
Debut
    Pour i de 0 à 99 faire
        TI[i] ← -1
    Finpour
    Pour i de 0 à N-1 faire
        TI[T[i]] ← T[i]
    Finpour
    j ← 0
    Pour i de 0 à 99 faire
        Si TI[i] ≠ -1 Alors
            T[j] ← TI[i]
            j ← j+1
        Finsi
    Finpour
```

- Placer chaque élément T[i] dans la case d'indice T[i] d'un tableau intermédiaire TI, sachant que les éléments du tableau TI sont initialisés à -1.
- Placer dans l'ordre tous les entiers différents de -1 du tableau TI, dans le

Exemple : pour le tableau T suivant :



Après application du principe de tri décrit précédemment, on obtient le tableau intermédiaire TI ci-dessous :

Objet	Nature / type
j, i	Entier
TI	tableau de 100 entier

Et on aura le tableau T trié suivant :

T	2	4	5	6	8	11
	0	1	2	3	4	5

Travail demandé :

Ecrire un programme qui permet de :

- Remplir un tableau T par N entiers positifs distincts et ne dépassant pas 99 avec $5 \leq N \leq 30$.
- Trier le tableau T en utilisant la méthode décrite ci-dessus.
- Afficher le tableau T après tri.

50 Minutes **7** Nouvelle méthode de tri

Analyse & Algorithme

Implémentation Avec PYTHON

Med Heni Frikha

Soit **T1** un tableau de **N** noms d'élèves **($4 \leq N \leq 20$)**. On suppose que le nom d'un élève est constitué de **10** lettres majuscules au maximum.
On se propose de trier les éléments de **T1** dans un tableau **T2** selon l'ordre croissant en utilisant le principe suivant :

1. chercher le nom qui a le plus petit poids dans **T1**, sachant que le poids d'un nom est la somme des codes ASCII des lettres qui le forment.
2. a) ranger ce nom dans **T2**.
   b) remplacer ce nom par "ZZZZZZZZZZ" dans **T1**.
3. répéter les étapes **1** et **2** sans tenir compte des noms remplacés par "ZZZZZZZZZZ" afin d'obtenir un tableau **T2** trié.

Pour i de 0 à N-1 faire
P ← PosMin(T1, N)
T2[i] ← T1[p]
T1[p] ← "ZZZZZZZZZZ"
Fin pour

Travail demandé

Ecrire un programme **Python** qui permet de saisir un entier **N ($4 \leq N \leq 20$)**, puis de remplir un tableau **T1** par **N** noms, de ranger les éléments de **T1** dans **T2** selon le principe décrit précédemment et d'afficher les éléments de **T2** ainsi que leurs poids.

Exemple :

T1	AMINE	KHALED	SALAH	IMED	ALI	HAFEDH
	0	1	2	3	4	5

Après Tri selon le poids, on obtient le tableau **T2** suivant :

T2	ALI	IMED	SALAH	AMINE	HAFEDH	KHALED

Le programme affiche :

ALI son poids = **214**
IMED son poids = **287**
SLAH son poids = **361**
AMINE son poids = **362**
HAFEDH son poids = **416**
KHALED son poids = **425**

Pour i de 0 à N-1 faire
P ← PosMin(T1, N)
T2[i] ← T1[p]
T1[p] ← "ZZZZZZZZZZ"
Fin pour

Correction de ....

Med Heni Frikha

Proc Saisir (N) → 4 ≤ N ≤ 20

Proc Remplir1 (T1, N) → T[i] formée par des lettres Maj

Proc Remplir2 (T2, T1, N) long(T[i]) ≤ 10

Proc Afficher (T2, N)

① Algorithme du Programme principal

```
Algorithme Sujet7
Debut
    Saisir (N)
    Remplir1 (T1, N)
    Remplir2 (T2, T1, N)
    Afficher (T2, N)
Fin
```

② Déclaration

T.D.O.G

objet	Nature / type
T1, T2	Tab
N	Entier
Saisir	} Procédure
Remplir1	
Remplir2	
Afficher	

T.D.N.T

Nouveau type
tab = tableau de 20 chaines

③ Algorithme des sous programmes

```
Procedure Saisir (@N: entier)
Debut
    Repeter
        Lire (N)
    Jusqu'a 4 ≤ N ≤ 20
Fin
```

Passage par variable / adresse / référence

Correction de ....

Med Heni Frikha

Passage par valeur

```
Procedure Remplir1 (@T1:tab, N:entier)
Debut
   Pour i de 0 à N-1 faire
      Repeter
         Lire(T1[i])
      jusqu'à long(T1[i])<5 et Verifier(T1[i])=Vrai
   Finpour
Fin
```

T.D.O.L

objet	Nature/Type
i	Entier
Verifier	Fonction/booléen

```
Fonction Verifier (ch:chaine): booléen
Debut
   test ← Vrai
   i ← 0
   tantque i<long(ch) et test=Vrai faire
      si "A" ≤ ch[i] ≤ "Z" Alors
         i ← i+1
      sinon
         test ← Faux
      Finsi
   Fintantque
   Retourner test
Fin
```

T.D.O.L

objet	N/T
i	Entier
test	booléen

Correction de ....

Med Heni Frikha

```
Fonction Verifier ( ch : chaine ) : booléen
Debut
    test ← Vrai
    i ← 0
    Repeter
        si "A" ≤ ch[i] ≤ "Z" Alors
            i ← i + 1
        sinon
            test ← Faux
        Finsi
    jusqu'a i >= long(ch) ou test = Faux
    Retourner test
Fin
```

T.D.O.L

objet	N/T
i	Entier
test	booléen

```
Fonction Poids ( ch : chaine ) : entier
Debut
    S ← 0
    pour i de 0 à long(ch) - 1 faire
        S ← S + ord(ch[i])
    Finpour
    Retourner S
Fin
```

T.D.O.L

objet	N/T
i, S	Entier
~~ch~~	~~chaine~~

on ne déclare pas les paramètres formels

Correction de ....

Med Heni Frikha

Le programme affiche :

ALI son poids = **214**
IMED son poids = **287**
SLAH son poids = **361**
AMINE son poids = **362**
HAFEDH son poids = **416**
KHALED son poids = **425**

```
Procedure Affiche (T2 :tab , N: entier)
Debut
   Pour i de 0 à N-1 faire
      Ecrire (T2[i], "Son poids = ", Poids (T2[i]))
   Finpour
Fin
```

T.D.O.L

objet	NIT
i	Entier
Poids	Fonction/Entier

```
Procedure Remplir (@ T2: tab, T1: tab , N: entier)
Debut
      Pour i de 0 à N - 1 faire
         P ← PosMin (T1, N)
         T2[i] ← T1[p]
         T1[p] ← "zzzzzzzzz"
      Finpour
Fin
Fonction Posmin (T: tab , N: entier) : entier
Debut
   Min ← Poids (T[0])
   P ← 0
   pour i de 1 à N-1 faire
      si Poids (T[i]) < min Alors
            min ← Poids (T[i])
            P ← i
      Fisi
   Finpour
Fin Retourner P
```

T.D.O.L

objet	NIT
i, P	Entier
Posmin	Fonction/Entier

T.D.O.L

objet	NIT
min, i, P	Entier
Poids	Fonction/Entier

Correction de ....

Med Heni Frikha

```
Fonction Posmin (T: tab, N: entier): entier
Debut
    P ← 0
    Pour i de 1 à N-1 faire
        Si Poids(T[i]) < Poids(T[P]) Alors
            P ← i
        Fsi
    Finpour
    Retourner P
Fin
```