Durée : 1h30

EMD 1 ALGORITHMIQUE

Exercice 1 (08 points)

Un nombre **semi-premier** est un nombre qui est le produit de **deux nombres premiers**. Exemple : 15 est un nombre semi-premier, car 15 = 3 * 5, et 3 et 5 sont premiers tous les deux.

On veut afficher les « ***n*** » premiers nombres semi-premiers :

- Ecrire l'énoncé explicite (l'analyse générale) qui permet de résoudre ce problème.
- Ecrire l'algorithme qui permet d'afficher les « ***n*** » premiers nombres semi-premiers.
- Ecrire le programme Pascal correspondant.

Exercice 2 (12 points)

Soit une matrice « A » de « n » lignes et « m » colonnes (n <= 100, m <= 100).

1. Ecrire une procédure « *LireMat* » qui permet de remplir la matrice.
2. Ecrire une fonction « *CompterZero* » qui calcule le nombre de zéros dans une colonne donnée.
3. Ecrire une procédure « *MinCol* » qui récupère le minimum de chaque colonne.
4. Ecrire une fonction « *NbCol* » qui permet de compter le nombre de colonnes contenant au moins un zéro.

Ecrire un algorithme qui permet à partir de la matrice « A »

- D'afficher le nombre de colonnes contenant au moins un zéro.
- D'afficher le numéro de la colonne contenant le minimum de la matrice « A ». (à égalité, donner le plus petit numéro de colonne).

NB : avant chaque procédure et fonction, veuillez expliquer, brièvement ou schématiquement, les paramètres et leurs types.

Soignez votre travail et bon courage !

Afud Iggerzan

Corrigé de l'EMD 1

Exercice 01 (08 points)

A. Analyse (02 points)

L'idée de base consiste à prendre un nombre « X » , tel que X = Divseur * Quot .
On pourra dire que « X » est semi premier si « Divseur » et « Quot » sont tous les deux premiers.
Pour cela :

- Donner N (« N » est le nombre de nombres semi premiers à afficher).
- Nsp = 0 (On initialise « Nsp » le compteur de nombres semi-premiers).
- On va faire varier X = 2, 3, 4, 5, 6, 7, … et à chaque fois :
  - ✓ On va chercher tous les diviseurs de « X »
    - *i.* Divseur = i (divseur est le diviseur).
    - *ii.* Quot = X div i (Quot est le quotient).
  - ✓ On va vérifier si « Divseur » est premier : Si *oui*, on va vérifier si « Quot » est lui aussi premier. Si Quot est lui aussi premier, alors on va incrémenter « Nsp » (on vient de trouver un nombre semi-premier).
  - ✓ On écrit le nombre semi-premier « X ».
  - ✓ On incrémente X. X= X+1 (on prend un nouveau X)
  - ✓ Et on s'arrête lorsque Nsp = *N* (on a trouvé et affiché les n premiers nombres semi-premiers)

B. Algorithme (4,5 points)

```
Algorithme Semi_premier ;
Var n, nsp, i, x, j, quot, diviseur : entier ;
Début
 Lire (n) ;
 Nsp ← 0 ;
 x ←2 ;
 Répéter
   Pour i allant de 2 à x div 2 faire
       Si (x mod i = 0) alors
          Diviseur ← i ;
          Quot ← x div I;
          j ← 2; {verification si diviseur est premier}
          Tant que (j<= (diviseur div 2)) et ((diviseur mod j) <> 0) faire
              j ← j+1 ;
         Fin Tant que ;
          Si (j > diviseur Div 2) alors
             j ← 2; {verification si quotient est premier}
             Tant que (j <= (quot div 2)) et ((quot mod j) <> 0) faire
                j ← j+1 ;
```

```
                Fin Tant que ;
                 Si (j › (Quot Div 2)) alors {compter x est Semi premier}
                    nsp ← nsp +1 ;
                   Ecrire ( Diviseur,' ',quot,' ', X) ;
                   Fin Si ;
                 Fin Si ;
              Fin Si ;
          Fin Pour ;
   x ← x +1 ;
 Jusqu'à (nsp = n) ;
Fin.
```

C. Programme en langage Pascal (1,5 points)

```
Program Semi_premier ;
Var
   n, nsp, i, x, j, quot, diviseur : integer ;
Begin
   readln (n) ;
   Nsp := 0 ;
   x := 2 ;
   Repeat
     For i := 2 To (x Div 2 ) Do
        Begin
          If (x Mod i = 0) Then
             Begin
               Diviseur := i ;
              Quot := (x Div i);
              j := 2;
               While (j <= (diviseur Div 2)) And ((diviseur Mod j) <> 0) Do
                  Begin
                    j := j+1 ;
                  End;

               If (j › diviseur Div 2) Then
                  Begin
                    j := 2; {verification si quotient est premier}
                     While (j <= (quot Div 2)) And ((quot Mod j) <> 0) Do
                        Begin
                          j := j+1 ;
                        End;
                     If (j › (Quot Div 2)) Then {compter xest Semi premier}
                        Begin
                          nsp := nsp +1 ;
                          writeln ( 'Diviseur = ', Diviseur,', quotion =  ',quot,'  nombre = ', x) ;
                        End;
                  End;
             End;
        End;
     x := x +1 ;
   Until (nsp = n) ;
End.
```

Exercice 02 (12 points)

1. Ecrire une procédure LireMat qui permet de remplir une matrice.

A. Description (0.25 point)

La procédure « LireMat » prend trois paramètres.

- La matrice A à remplir. Elle est de type « Matrice », déclaré dans l'algorithme principal. La matrice A doit être introduite à la procédure « LireMat » en entrée/sortie (E/S), puisque ses valeurs seront modifiées et renvoyées à l'algorithme appelant.
- Le nombre de lignes « n » de la matrice A. Il est de type entier et introduit en entrée (E) à la procédure.
- Le nombre de colonnes « m » de la matrice A. Il est de type entier et introduit en entrée (E) à la procédure.

Nous avons besoin de deux variables locales, de type entier, dans « LireMat » pour parcourir les cases de la matrice A. Soient ces deux variables locales « i » et « j ».

B. Procédure (0.75 point)

La procédure « LireMat » est donnée comme suit :

```
Procedure LireMat (E/S A : Matrice ; E n : entier ;  E m : entier) ;
Var i, j : entier ;
Début
  Pour i allant de 1 à n faire
    Pour j allant de 1 à m faire
      Ecrire ('Donnez la valeur de l' élément ', i ,' ,', 'j) ;
      Lire (M[i,j]) ;
  Fin Pour ;
Fin Pour ;
Fin ;
```

2. Ecrire une fonction CompterZero qui calcule le nombre de zéros dans une colonne donnée.

A. Description (0.5 point)

La fonction « CompterZero » prend trois paramètres, et retourne un résultat de type entier. Les paramètres de la fonction « CompterZero » sont :

- La matrice « A » qui contient la colonne donnée. Elle est de type « Matrice » déclaré dans l'algorithme principal.
- Le nombre de lignes « n » de la matrice « A ». Il est de type entier.
- L'indice « C » de la colonne à vérifier dans la matrice « A ». Il est de type entier.

La fonction « CompterZero » utilise deux variables locales, à savoir : « i », de type entier, pour parcourir les éléments de la colonne « C », et « Nbr_zero », de type entier, pour compter le nombre de zéros de la colonne « C ».

C. Fonction (1.5 points)

La fonction « CompterZero » est donnée comme suit :

```
Fonction CompterZero (A : Matrice ; n : entier ; C : entier) : entier ;
Var  Nbr_zero, i : eniter ;
Début
 Nbr_zero = 0 ;
 Pour i allant de 1 à n faire
   Si (A[i,C] = 0) alors
      Nbr_zero ← Nbr_zero + 1 ;
   Fin Si ;
 Fin Pour ;
 CompterZero ← Nbr_zero ;
Fin ;
```

3. Ecrire une procédure MinCol qui récupère le minimum de chaque colonne.

A. Description (0.5 point)

La procédure « MinCol » prend trois paramètres en entrée (E).

- La matrice A à vérifier. Elle est de type « matrice », déclaré dans l'algorithme principal.
- Le vecteur V pour récupérer le minimum de chaque colonne de « A ». Il est de type « Vecteur », déclaré dans l'algorithme principal.
- Le nombre de lignes « n » de la matrice « A ». Il est de type entier.
- Le nombre de colonnes « m » de la matrice « A ». Il est de type entier.

Nous avons besoin de trois variables locales. Les variables : « i », « j », de type entier, parcourir les cases de la matrice « A ». Et, La variable « Min » pour stocker la valeur minimale lors du traitement d'une colonne.

B. Procédure (1.5 points)

La procédure « MinCol » est donnée comme suit :

```
Procédure MinCol (E  A : Matrice ; E n : entier ; E m : eniter ; E /S V : Vecteur) ;  (0.5 point)
Var i, j, Min : entier ;
Début
 Pour j allant de 1 à m faire
   Min ←A [1,j] ;
   Pour i allant de 2 à n faire
    Si (A [i,j] < Min ) alors
      Min ← A [i,j] ;
    Fin Si ;
  Fin Pour ;
  V[j] ← Min ;
 Fin Pour ;
Fin ;
```

4. Ecrire une fonction « NbCol » qui permet de compter le nombre de colonnes contenant au moins un zéro.

Description (0.5 points)

La fonction « NbCol » prend trois paramètres, et retourne un résultat de type entier. Les paramètres de la fonction « NbCol » sont :

- La matrice « A » à vérifier. Elle est de type « matrice » déclaré dans l'algorithme principal.
- Le nombre de lignes « n » de la matrice « A ». Il est de type entier.
- L'indice « C » de la colonne à vérifier dans la matrice « A ». Il est de type entier.

La fonction « NbCol » utilise trois variables locales, à savoir : « i » et « j », de type entier, pour parcourir les éléments de la matrice « A ». Et « Nbr », de type entier, pour compter le nombre des colonnes de « A » ayant au moins un zéro.

Fonction (1.5 points)

La fonction « « NbCol » » est donnée comme suit :

```
Fonction NbCol (A : matice ; n : entier ;
m : entier) : entier ; (0.5 point)
Var
        i, j, Nbr : entier ;
        Trouve : booléen ;
Début
   Nbr ←0 ;
   Pour j allant de 1 à m faire
      Trouve ←faux ;
      i ←1 ;
      Tant que (i <= n) and (Trouve = faux)
faire
          Si (M [i,j] = 0) alors
             Trouve ← vrai ;
          Fin Si ;
       Si (Trouve = vrai) alors
          Nbr ←Nbr +1 ;
       Fin Si ;
  Fin Pour ;
  NbCol ←Nbr ;
Fin ;
```

```
Fonction NbCol (A : matice ; n : entier ;
m : entier) : entier ;
Var
        i, j, Nbr : entier ;
        Trouve : booléen ;
Début
   Nbr ←0 ;
   Pour j allant de 1 à m faire
          Si (CompterZero (A, n, j) > 0) alors
             Nbr ←Nbr +1 ;
          Fin Si ;
   Fin Pour ;
   NbCol ←Nbr ;
Fin ;
```

5. Ecrire un algorithme qui permet à partir de la matrice « A » :
   - D'afficher le nombre de colonnes contenant au moins un zéro.
   - D'afficher le numéro de la colonne contenant le minimum de la matrice A. (A égalité, donner le plus petit numéro de colonne).

A. Description

Pour réaliser l'algorithme demandé nous avons besoins d'un nouveau type « Matrice », de quelques variables globales, de quelques sous-programmes parmi ceux que nous avions décrits ci-haut.

- **Type :** « Matrice =Tableau [1..100, 1..100] de entier », pour pouvoir passer une matrice en paramètre des sous-programmes. « Vecteur : Tableau [1..100] de entier », pour
- **Variables globales :** Une matrice « A » déclarée en utilisant le type Matrice. Deux entier « n » et « m » pour spécifier la taille de la matrice « A ». Un entier « Nombre » pour récupérer le nombre de colonnes de « A » contenant au moins un zéro. Et enfin, un entier indice pour récupérer le numéro de la colonne contenant le minimum de « A ».
- **Procédures :** La procédure « LireMat » pour remplir la matrice « A ». La procédure «MinCol » pour compter le nombre de colonnes de « A » contenant au moins un zéro.
- **Fonction :** La fonction « NbCol » pour trouver le numéro de la colonne contenant le minimum de la matrice « A ».

```
Algorithme exo2 ; (La forme générale 0.25 point)
Type (0.5 point)
        Matrice = Tableau [1...100, 1...100] de entier ;
        Vecteur = Tableau [1..100] de entier ;
Var
        n, m, Nombre, Min, Indice : entier ;
        A : Matrice ;
        V : Vecteur ;

Procédure LireMat (E/S A : Matrice ; E n : entier ; E m : entier) ;
  Var i,j : entier ;
  Debut
    Pour i allant de 1 à n faire
      Pour j allant de 1 à m faire
       Ecrire ('Donnez la valeur de l' élément ', i ,' ,', 'j) ;
       Lire (M [i,j]) ;
      Fin Pour ;
    Fin Pour ;
Fin ;

Fonction NbCol (A : Matrice ; n : entier ; m : entier) : entier ;

Var
      i, j, Nbr : entier ;
      Trouve : booléen ;

Début
  Nbr ←0 ;
  Pour j allant de 1 à m faire
    Trouve ←faux ;
    i←1 ;
```

```
      Tant que (i<= n) and (Trouve = faux) faire
         Si (M [i,j] = 0) alors
            Trouve ← vrai ;
         Fin Si ;
      Si (Trouve = vrai) alors
         Nbr ←Nbr +1 ;
      Fin Si ;
   Fin Pour ;
Fin ;

Procédure MinCol (E  A : Matrice ; E n : entier ; E m : eniter ; E /S V : Vecteur) ;
Var i, j, Min : entier ;
Début
  Pour j allant de 1 à m faire
    Min ←A [1,j] ;
    Pour i allant de 2 à n faire
     Si (A [i,j] < Min ) alors
        Min ← A [i,j] ;
     Fin Si ;
   Fin Pour ;
   V[j] ← Min ;
 Fin Pour ;
Fin ;
Début
  Ecrire ('Donnez le nombre des lignes de la matrice') ;
  Lire (n) ;
  Ecrire ('Donnez le nombre des colonnes de la matrice') ;
  Lire (m) ; (Lecture de n et m (0.25 point))
  LireMat (A, n, m) ;  (0.5 point)
  Nombre ← NbCol (A, n, m) ; (0.5 point)
  Ecrire ('Le nombre de colonnes qui contiennent au minimum un zéro est ', Nombre) ; (0.25 point))
  MinCol (A, n, m, V); (0.5 point)
  Min ← V [1] ;
  Pour j allant de 1 à m faire  (1 point)
     Si (Min < V[j])  alors
          Min ← V[j]) ;
          Indice ← j ;
     Fin Si ;
  Ecrire ('Le numéro de la colonne contenant la valeur minimale de la matrice A est ', Indice) ; (0.25 point))
Fin.
```