Groupe5:386

Nom et prenom	groupe
BALL Mohamed	386N
BENDIMERAD WAHIB	386N
CATTIN SOLENN	386N
DIALLO Fatoumata	386N
HISSEINE MBODOU KOKIYA	386N

UNIVERSITÉ DE NANTES

Informatique fondamentale

Distanciel: Suites “linéarisantes” et “partitionnantes”

L2 informatique
2021-2022

Première partie

Soit B une structure binaire, on a que B(1) est un ensemble et que B(2) ⊆ B(1) × B(1)

sous partie1

1. Reflexive(B) :fonction qui prend en entrée une structure binaire et retourne le booléen vrai si et seulement si la structure binaire B est réflexive.

```
Reflexive :=func(B);
        local x;
         if(B(1) = {}) then return true;
         else
            take x from B(1);
            return [x,x] in B(2) and Reflexive(B);
         end if;
end func;
```

exécution :

Donnée(s)	Résultat(s)
A1:={1,2,3}; A2:={[1,2],[1,3],[1,1],[2,2],[3,3]}; A:=[A1,A2]; Reflexive(A);	true;

2-AntiReflexive(B) :fonction qui prend en entrée une structure binaire et retourne le booléen vrai si et seulement si la structure binaire B est antiréflexive.

```
AntiReflexive:=func(B);
   local x;
         if(B(1) = {}) then return true;
         else
            take x from B(1);
            return not [x,x] in B(2) and AntiReflexive(B);
          end if;
end func;
```

exécution :

Donnée(s)	Résultat(s)
A1:={1,2,3}; A2:={[1,2],[1,3],[1,1],[2,2],[3,3]}; A:=[A1,A2]; A3:={[1,2],[1,3]}; C:=[A1,A3];	AntiReflexive(A); false; AntiReflexive(C); true;

3-Symetrique(B):fonction qui prend en entrée une structure binaire et retourne le booléen vrai si et seulement si la structure binaire B est symétrique.

```
Symetrique := func(B);
   if(B(2) = {}) then return true;
   else
         take x from B(2);
        return [x(2),x(1)] in B(2) and Symetrique([B(1),B(2) less [x(2),x(1)]]);
   end if;
end func;
```

exécution :

Donnée(s)	Résultat(s)
A1:={1,2,3}; A2:={[1,2],[2,1],[1,3],[3,1]}; A:=[A1,A2]; Symetrique(A);	true;

4-AntiSymetrique(B) : fonction qui prend en entrée une structure binaire et retourne le booléen vrai si et seulement si la structure binaire B est antisymétrique.

```
AntiSymetrique := func(B);
         if(B(2) = {}) then return true;
          else
           take x from B(2);
           return x(1)=x(2) or not [x(2),x(1)] in B(2) and AntiSymetrique(B) ;
          end if;
end func;
```

exécution :

Donnée(s)	Résultat(s)
A1:={1,2,3}; A2:={[1,1],[2,2],[3,1]}; A:=[A1,A2]; AntiSymetrique(A);	true;

5-Transitive(B) : fonction qui prend en entrée une structure binaire et retourne le booléen vrai si et seulement si la structure binaire B est Transitive.

```
Transitive:=func(B);
   if(B(2) = {}) then return true;
   else
      take x  from B(2);
      take y  from B(2);
      return x(2)=y(1) and [x(1),y(2)] and Transitive([B(1),B(2) less [x(1),y(2)]]);
end if;
end func;
```

exécution :

Donnée(s)	Résultat(s)
A1:={1,2,3}; A2:={[1,2],[2,3],[1,3]}; A:=[A1,A2]; Transitive(A);	true;

sous partie2

On appelle pseudo-chemin, dans une structure binaire non vide B, toute suite finie non vide d'éléments de B(1) telle que deux éléments consécutifs de cette suite induisent un élément de B(2).On appelle chemin élémentaire dans une structure binaire non vide B tout pseudo-chemin uniquement constitué d'éléments distincts.Une structure binaire non vide B est sans circuit si tous ses pseudo-chemins sont des chemins élémentaires.On peut montrer qu'une structure binaire non vide est sans circuit si et seulement s'il existe $k \in N^*$ et une suite (X_i) $i \in [k]$, dite suite "linéarisante", telle que :
1. $\{X_i : i \in [k]\}$ soit une partition de B(1),
2. $\forall e \in B(2), \exists i, j \in [k]$ tels que $i <_N j$, $e(1) \in X_i$ et $e(2) \in X_j$.

1-Fonction qui prend une structure binaire non vide en paramètre et retourne une suite "linéarisante" de longueur maximale si la structure binaire est sans circuit et elle retourne la suite vide sinon.

```
Linearisantemax  := func (B);
    X := [e : e in pow(B(1)) : #e = 1];
     if(Symetrique(B)=true or AntiReflexive(B)=false) then return ([]);
     else
     return line_rec(B, X);
   end if;
end func;
```

```
$line_rec:fonction qui permet de mettre les elements de X dans  le bon ordre
line_rec := func(B, X);
   if (B(2) /= {}) then
      take e from B(2);
      i :=  arb({ n : n  in {1..#X} : X(n) = {e(1)}});
      j :=  arb({ m : m  in {1..#X} : X(m) = {e(2)}});
      if (i >= j) then
         tmp := X(j);
         X(j) := X(i);
         X(i) := tmp;
      end if;
      return line_rec(B, X);
   else
      return X;
   end if;
end func;
```

exécution :

Donnée(s)	Résultat(s)
A1:={1,2,3}; A2:={[1,2],[3,2]}; A:=[A1,A2]; Linearisantemax(A);	[{3}, {1}, {2}];

2-Fonction qui prend une structure binaire non vide en paramètre et retourne une suite "linéarisante" de longueur minimale si la structure binaire est sans circuit et elle retourne la suite vide sinon.

```
Linearisantemin := func(B);
          X:=[0*e : e in {1..#B(1)}];
          x := line_rec(B, x);
           tmp := minmax(x);
           min_x  := tmp(1);
           max_x  := tmp(2);
          p := {{e : x(e) = i} : i in {min_x..max_x}};
           return p;
```

```
end func;
$fonctions qui permettent de mettre les éléments de X dans  le bon ordre
line_rec := func(B, X);
   if (B(2) /= {}) then
      take e from B(2);
         if (X(e(1)) >= X(e(2))) then
         X(e(1)) := X(e(1))-1;
         end if;
      return line_rec(B, X);
   else
      return X;
end if
end func;

minmax :=  func(X);
          min_x := 0;
          max_x := 0;
return minmax_rec(X,min_x,max_x,1);
end func;
minmax_rec := func(X,min_x,max_x,i);
         if(i > #X) then return [min_x,max_x];
         else
         if (X(i) > max_x) then return minmax_rec(X,min_x,X(i),i+1);
         end if;
         if (X(i) < min_x) then return minmax_rec(X,X(i),max_x,i+1);
         end if;
         return minmax_rec(X,min_x,max_x,i+1);
         end if;
end func;
```

exécution :

Donnée(s)	Résultat(s)
A1:={1,2,3}; A2:={[1,2],[3,2]}; A:=[A1,A2]; Linearisantemin(A);	[{1,3}, {2}];

3-

a-Montrons qu'une structure binaire non vide B est sans circuit si et seulement si elle possède une suite "linéarisante".

preuve par l'absurde: Posons B(1)={1,2}; B(2)={(1,2),(2,1)}; X=({1},{2})

$1 \in X_i$ et $2 \in X_j$ tel que $i<j$ condition vérifié selon (1,2). Mais $j<i$ selon (2,1) on a donc une contradiction.

Conclusion:une structure binaire non vide B est sans circuit si et seulement si elle possède une suite "linéarisante".

b-la longueur maximale d'une suite "linéarisante" est égale au cardinal de B(1).La suite produite par notre fonction Linearisantemax(.) est bien une suite "linéarisante" de longueur maximale car lorsqu'on l'exécute elle nous renvoie une suite dont la taille est égale au cardinal de B(1), ses composantes forment une partition de B(1) et pour chaque relation de B(2) $i<j$.

Le caractère maximal de la longueur n'assure pas l'unicité car :

posons B(1)={1,2,3,4}et B(2)={(1,2),(3,4)}.Soient X1,X2 deux suites linéarisantes telles que:

X1=({1},{2},{3},{4}); X2=({1},{3},{2},{4}).Les deux suites vérifient les propriétées de suite "linéarisante" et elles sont de longueurs maximales Ainsi on a prouver que Le caractère maximal de la longueur n'assure pas l'unicité.

c-la longueur minimale d'une suite "linéarisante" peut être égal à 2. La suite produite par notre fonction Linearisantemin(.) est bien une suite "linéarisante" de longueur minimale car lorsqu'on l'exécute elle nous renvoie bien une suite avec la longueur la plus petite qu'on peut obtenir telle que toutes les conditions soient vérifiés.
Le caractère minimal de la longueur n'assure pas l'unicité car :
posons B(1)={1,2,3}et B(2)={(1,2)}.Soient X1,X2 deux suites linéarisantes telles que:
X1=({1,3},{2}); X2=({1},{2,3}).Les deux suites sont de longueurs minimale Donc en conclusion Le caractère minimal de la longueur n'assure pas l'unicité.

Deuxième partie

Soit n ∈ N, pour rappel, le nombre de bijections entre deux ensembles de n éléments est n!.
1. Explication du cas n = 0.
Soient A et B deux ensembles tous deux de n éléments. Pour n=0, on a l'application bijective f: A → B Or A={} et B={}, donc il y a une unique association de A vers B qui est f=({}, {}).

2.l'égalité ensembliste qui relie les ensembles EnsBij(A, B) et EnsBij(A \ {a}, B \ {b}).
On sait que #(AxB) = #A * #B, #EnsBij(A, B) est donc le cardinal du produit cartésien entre un ensemble de taille #B et EnsBij(A\{a}, B\{b}), a ∈ A et b ∈ B.

2.2)preuve par récurrence du fait que #EnsBij(A, B) = (#A)!.
P(n): ∃A,B deux ensembles de taille n, on cherche a montrer que #EnsBij(A,B) = (#A)!.
Initialisation pour n=1
#EnsBij(A,B) = 1
(#A)! = 1
donc P(1) est vraie.
Schéma inductif
Soit n ∈ ℕ* tel que P(n-1) vraie, montrons P(n).
Pour a ∈ A et b ∈ B, on a :
#EnsBij(A,B) = #B * #EnsBij(A\{a}, B\{b})
Donc, pour #A=#B=1
(#A)! = #B * #EnsBij(A\{a}, B\{b})
Or #(A\{a}) = #(B\{b}) = n-1
Donc #EnsBij(A\{a}, B\{b}) = (n-1)!
Or #A = #B = n
Donc (n-1)! * #B = (n-1)! * n = n!

Alors #EnsBij(A,B) = (#A)!
Donc P(n) est vraie.
Conclusion
Par le théorème de récurrence, ∀n∈ℕ*, ∃A,B deux ensembles de taille n, P(n) vraie.

2.3)EnsBij(., .):fonction récursif prenant comme paramètres
d'entrée deux ensembles A et B de même cardinal, et retournant l'ensemble des bijections de A dans B

```
$fonction qui retourne une bijection
EnsBij := func (A,B);
   local t;
   local l;
        if(A={}) then return {};
        else
        take t from A;
        take l from B;
        X:={[t,l]};
        return  X + EnsBij(A,B) ;
        end  if;
        end func;
```

exécution :

Donnée(s)	Résultat(s)
A:={1,2,3,4}; B:={9,8,7,6}; EnsBij(A,B);	{[3, 6], [4, 8], [2, 9], [1, 7]};

Troisième partie

Remarque: on n'arrive pas à représenter 'j' de tel sorte qu'il ne soit pas au même niveau que 'i'.

Une suite de suites d'ensembles est une structure du type $((L_{ij})\ j\in J_i)\ i\in I$ où les L_{ij} sont des ensembles,elle est partitionnante pour X si $\{L_{ij} : i \in I, j \in J_i\}$ est une partition de l'ensemble X.Soient (X_1 , L_1) et (X_2 , L_2) deux couples tels que, pour tout k dans [2], $L_k = ((L_{k,ij})\ j\in J_{k,i})\ i\in I_k$ soit une suite de suites d'ensembles partitionnante pour X_k.Une bijection φ de l'ensemble X_1 dans l'ensemble X_2 est dite(L_1,L_2)-compatible si $\forall i \in I_1, \forall j \in J_{1,i}$ la restriction de φ à $L_{1,ij}$ est une bijection de $L_{1,ij}$ dans $L_{2,ij}$.

1-les conditions à vérifier entre deux couples (X_1 , L_1) et (X_2 , L_2) pour qu'il puisse exister des bijections (L_1 , L_2)-compatibles de X_1 dans X_2 : $(\#L_{1,ij})= (\#L_{2,ij})$ et $(\#X_1)= (\#X_2)$.

2- Expression, en fonction des $L_{1,ij}$, le nombre de bijections compatibles de X_1 dans X_2 :

$$\sum_{i\in I_1} \sum_{j\in J_{1,i}} (\#L_{1,ij})! \ .$$

3-

4-le cardinal de l'ensemble des bijections X_1 dans X_2 qui soient (L_1 , L_2)-compatibles de ({1, 2, 3, 4, 5, 6}, [[{1, 2}, {3, 4, 5, 6}], [{7, 8, 9}, {10, 11, 12}]]) et ({'a','b','c','d', 'e','f '}, [[{'a','b'}, { 'c','d','e','f'}], [{'g','h','i'}, {'j','k','l'}]]) est égal à 8!.