Concepts Informatiques

2018–2019

Matthieu Picantin

Tests et examens

- CC : résultat des 3 tests (ou plus) effectués en TD
- E0 : partiel (résultats et copies corrigées sur moodle)
- E1 : **examen le 18 avril à 12h00**
- E2 : examen mi-juin

Notes finales

- Note session 1 : 20% CC + 20% E0 + 60% E1
- Note session 2 : max(E2, 20% CC + 80% E2)

Rappel

pas de note $\Longrightarrow$ pas de moyenne $\Longrightarrow$ pas de semestre

`moodlesupd.script.univ-paris-diderot.fr`

```
int res=1,cpt=2,arg=7;
while(cpt<=arg) res*=cpt++;
return res;
```

pensée

calcul
récursion
fonction
objet
⋮

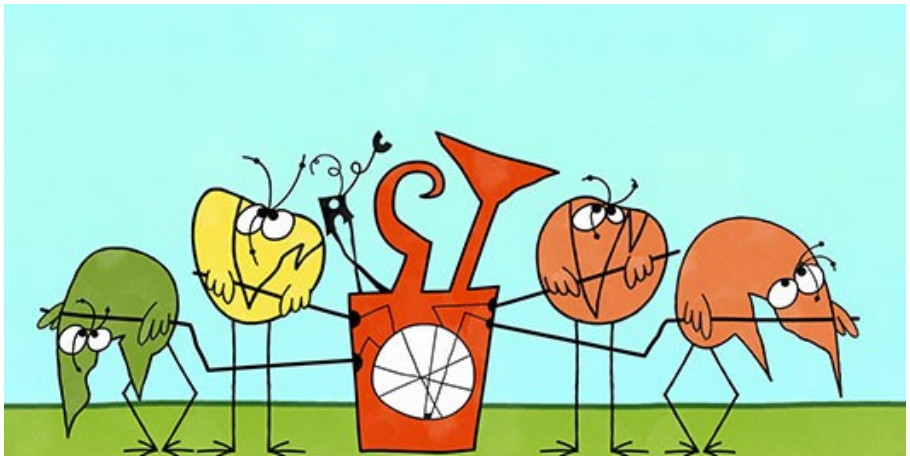
machine

circuit
pile
registre
mémoire
⋮

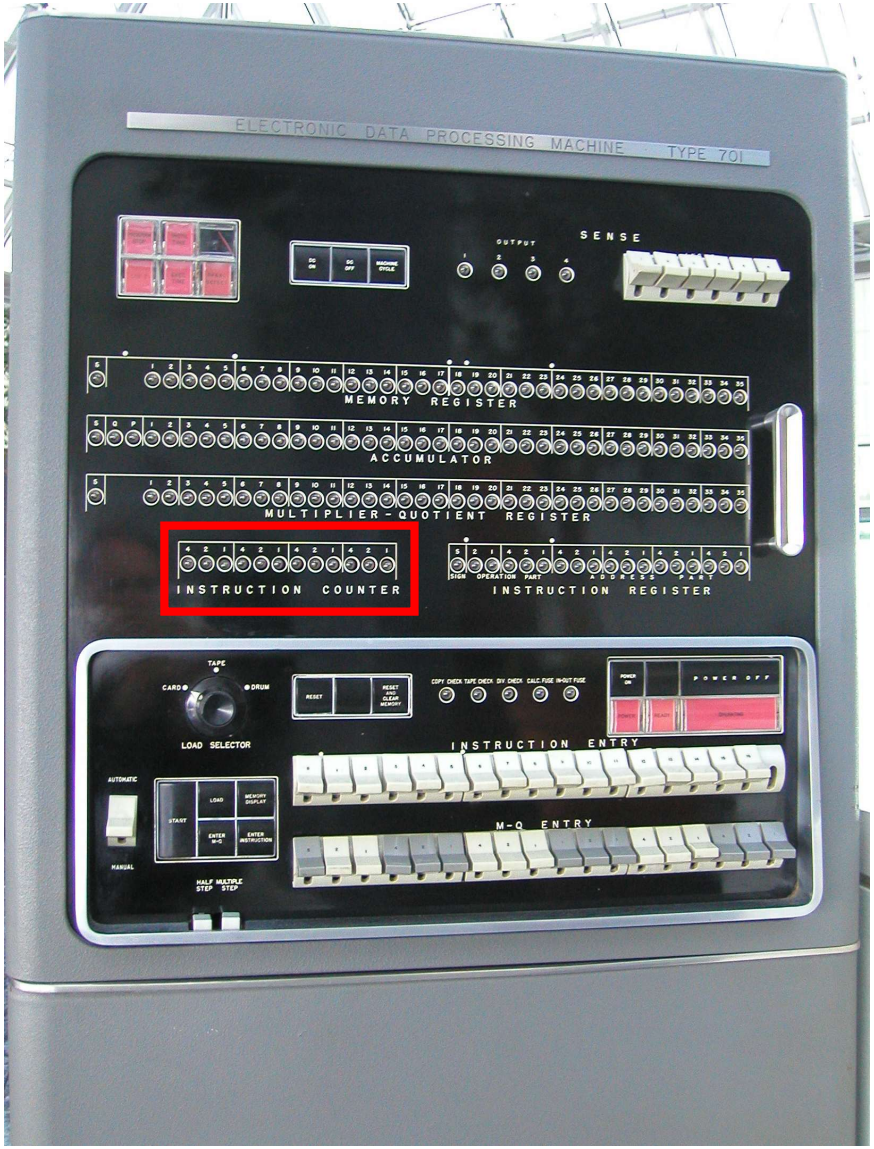
```
10111000 00000001 00000000
00000000 00000000 10111010
00000010 00000000 00000000
00000000 00111001 11011010
01111111 00000110 00001111
10101111 11000010 01000010
11101011 11110110 11000011
```

Traduire tout programme
dans une forme très proche
de celle acceptée par les machines

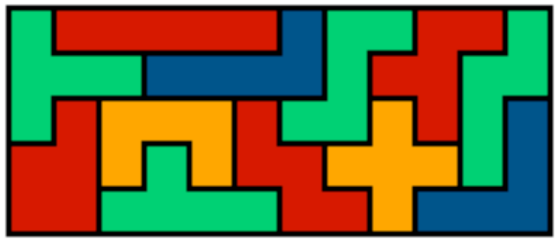
```
class Recursion{
    static int u(int m, int n){
        if(m==0 || n==0) return 1;
        return 2 * u(m-1, n+1);
    }
    public static void main(String[] a){
        System.out.println(u(3,7));
    }
}
```

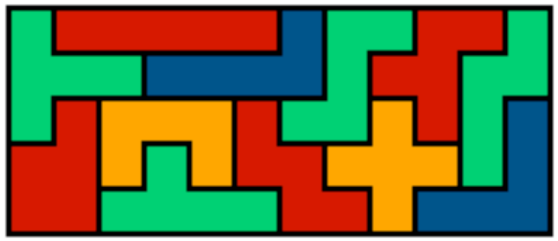


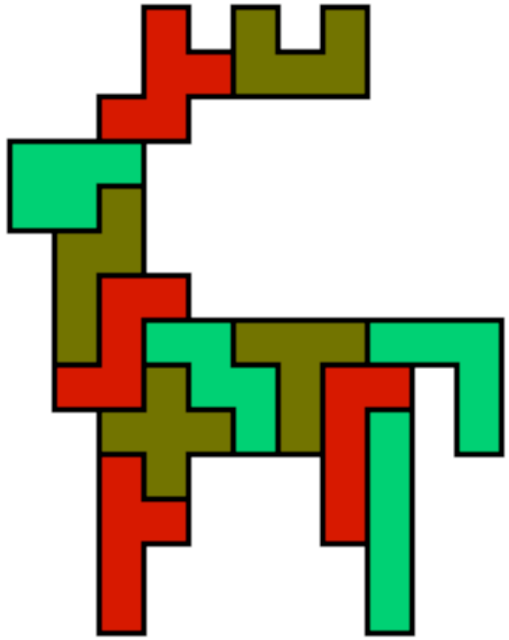
```
class RecursionTraduit{
   public static void main(String[] a){
       Stack<Bloc> p=new Stack<Bloc>();
       int ic=0;
       Bloc b;
       while(true){
            System.out.println(ic+"␣"+p);
            switch(ic++){
                case 0: p.push(new Bloc2(ic,3,7)); ic=100; break;
                case 1: System.out.println(((BlocR)(p.pop())).val); break;

                case 100: if(!( ((Bloc2)(p.peek())).arg1==0
                              || ((Bloc2)(p.peek())).arg2==0 )) ic+=2; break;
                case 101: ((BlocR)(p.peek())).val=1; break;
                case 102: ic=((BlocR)(p.peek())).adr; break;
                case 103: p.push(new Bloc2(ic,
                                  ((Bloc2)(p.peek())).arg1-1,
                                  ((Bloc2)(p.peek())).arg2+1)); ic=100; break;
                case 104: b=p.pop(); ((BlocR)(p.peek())).val=2 * ((BlocR)b).val; break;
                case 105: ic=((BlocR)(p.peek())).adr; break;

                case 200: if(!( ((Bloc1)(p.peek())).arg1==0)) ic+=2; break;
                case 201: ((BlocR)(p.peek())).val=1; break;
                case 202: ic=((BlocR)(p.peek())).adr; break;
                case 203: if(!( ((Bloc1)(p.peek())).arg1%2==0 )) ic+=3; break;
                case 204: p.push(new Bloc1(ic,((Bloc1)(p.peek())).arg1/2)); ic=200; break;
                case 205: b=p.pop(); ((BlocR)(p.peek())).val
                                =((Bloc1)(p.peek())).arg1 * ((BlocR)b).val; break;
                case 206: ic=((BlocR)(p.peek())).adr; break;
                case 207: p.push(new Bloc1(ic,((Bloc1)(p.peek())).arg1/2)); ic=200; break;
                case 208: b=p.pop(); ((BlocR)(p.peek())).val=((BlocR)b).val; break;
                case 209: ic=((BlocR)(p.peek())).adr; break;

                case 1000: if(!( ((Bloc3)(p.peek())).arg1==0
                              || ((Bloc3)(p.peek())).arg2==0 )) ic+=2; break;
                case 1001: ((BlocR)(p.peek())).val=((Bloc3)(p.peek())).arg3; break;
                case 1002: ic=((BlocR)(p.peek())).adr; break;
                case 1003: p.push(new Bloc3(ic,((Bloc3)(p.peek())).arg1-1,
                                               ((Bloc3)(p.peek())).arg2+1,
                                            2*((Bloc3)(p.peek())).arg3)); ic=1000; break;
                case 1004: b=p.pop(); ((BlocR)(p.peek())).val=((BlocR)b).val; break;
                case 1005: ic=((BlocR)(p.peek())).adr; break;

                case 1500: p.push(new Bloc3(ic,((Bloc2)(p.peek())).arg1,
                                               ((Bloc2)(p.peek())).arg2,1)); ic=1000; break;
                case 1501: b=p.pop(); ((BlocR)(p.peek())).val=((BlocR)b).val; break;
                case 1502: ic=((BlocR)(p.peek())).adr; break;

                default: System.exit(0);
            }
        }
    }
}
```

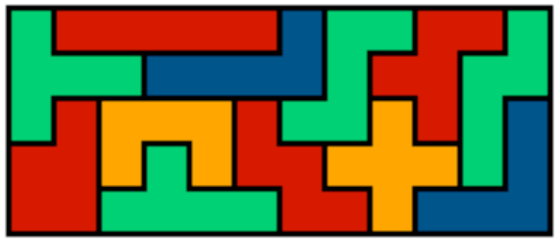


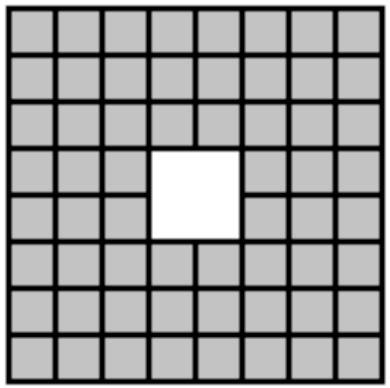

```
class RecursionTraduit{
   public static void main(String[] a){
       Stack<Bloc> p=new Stack<Bloc>();
       int ic=0;
       Bloc b;
       while(true){
           System.out.println(ic+"␣"+p);
           switch(ic++){
               case 0: p.push(new Bloc2(ic,3,7)); ic=100; break;
               case 1: System.out.println(((BlocR)(p.pop())).val); break;

               case 100: if(!( ((Bloc2)(p.peek())).arg1==0
                           || ((Bloc2)(p.peek())).arg2==0 )) ic+=2; break;
               case 101: ((BlocR)(p.peek())).val=1; break;
               case 102: ic=((BlocR)(p.peek())).adr; break;
               case 103: p.push(new Bloc2(ic,
                                   ((Bloc2)(p.peek())).arg1-1,
                                   ((Bloc2)(p.peek())).arg2+1)); ic=100; break;
               case 104: b=p.pop(); ((BlocR)(p.peek())).val=2 * ((BlocR)b).val; break;
               case 105: ic=((BlocR)(p.peek())).adr; break;

               case 200: if(!( ((Bloc1)(p.peek())).arg1==0)) ic+=2; break;
               case 201: ((BlocR)(p.peek())).val=1; break;
               case 202: ic=((BlocR)(p.peek())).adr; break;
               case 203: if(!( ((Bloc1)(p.peek())).arg1%2==0 )) ic+=3; break;
               case 204: p.push(new Bloc1(ic,((Bloc1)(p.peek())).arg1/2)); ic=200; break;
               case 205: b=p.pop(); ((BlocR)(p.peek())).val
                              =((Bloc1)(p.peek())).arg1 * ((BlocR)b).val; break;
               case 206: ic=((BlocR)(p.peek())).adr; break;
               case 207: p.push(new Bloc1(ic,((Bloc1)(p.peek())).arg1/2)); ic=200; break;
               case 208: b=p.pop(); ((BlocR)(p.peek())).val=((BlocR)b).val; break;
               case 209: ic=((BlocR)(p.peek())).adr; break;

               case 1000: if(!( ((Bloc3)(p.peek())).arg1==0
                           || ((Bloc3)(p.peek())).arg2==0 )) ic+=2; break;
               case 1001: ((BlocR)(p.peek())).val=((Bloc3)(p.peek())).arg3; break;
               case 1002: ic=((BlocR)(p.peek())).adr; break;
               case 1003: p.push(new Bloc3(ic,((Bloc3)(p.peek())).arg1-1,
                                              ((Bloc3)(p.peek())).arg2+1,
                                          2*((Bloc3)(p.peek())).arg3)); ic=1000; break;
               case 1004: b=p.pop(); ((BlocR)(p.peek())).val=((BlocR)b).val; break;
               case 1005: ic=((BlocR)(p.peek())).adr; break;

               case 1500: p.push(new Bloc3(ic,((Bloc2)(p.peek())).arg1,
                                              ((Bloc2)(p.peek())).arg2,1)); ic=1000; break;
               case 1501: b=p.pop(); ((BlocR)(p.peek())).val=((BlocR)b).val; break;
               case 1502: ic=((BlocR)(p.peek())).adr; break;

               default: System.exit(0);
           }
       }
   }
}
```

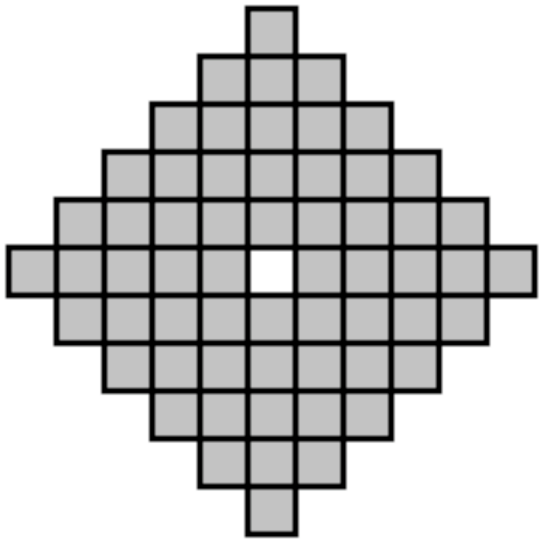


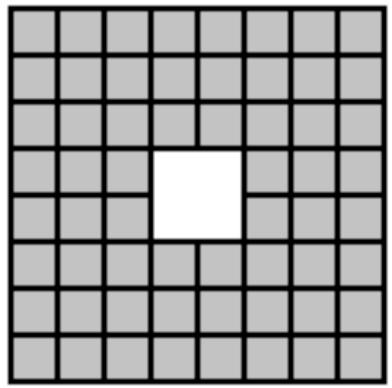












Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

$$\begin{matrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}$$

Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

$$\begin{matrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}$$

Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

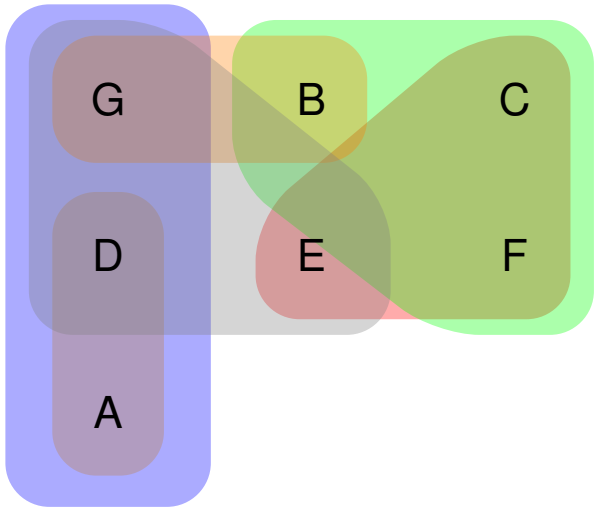
Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

$$\begin{matrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}$$

Couverture exacte (version graphique)

Trouver une sélection de couvertures
couvrants exactement chacun des points



Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Algo X de Knuth (version graphique)

s'il n'y aucun point, renvoyer $\varnothing$
choisir un point c (peu couvert)
| choisir une couverture ℓ couvrant c
| et l'inclure dans la solution partielle
| | supprimer les points couverts par ℓ
| | et les couvertures couvrant ces points
| | lancer la récursion sur le graphe réduit

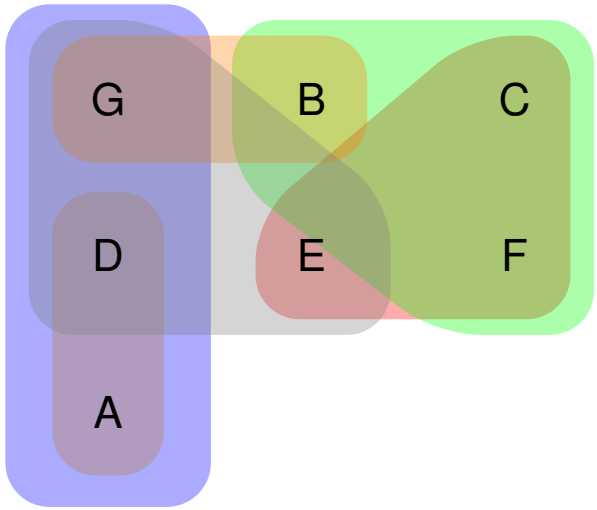
Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

$$\begin{matrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}$$

Couverture exacte (version graphique)

Trouver une sélection de couvertures
couvrants exactement chacun des points



Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Algo X de Knuth (version graphique)

s'il n'y aucun point, renvoyer $\varnothing$
choisir un point c (peu couvert)
| choisir une couverture ℓ couvrant c
| et l'inclure dans la solution partielle
| | supprimer les points couverts par ℓ
| | et les couvertures couvrant ces points
| | lancer la récursion sur le graphe réduit

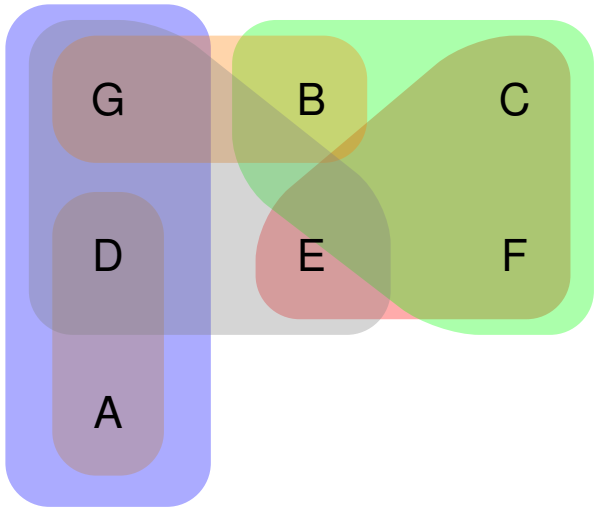
Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

A	B	C	D	E	F	G
0	0	1	0	1	1	0
1	0	0	1	0	0	1
0	1	1	0	0	1	0
1	0	0	1	0	0	0
0	1	0	0	0	0	1
0	0	0	1	1	0	1

Couverture exacte (version graphique)

Trouver une sélection de couvertures
couvrants exactement chacun des points



Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Algo X de Knuth (version graphique)

s'il n'y aucun point, renvoyer $\varnothing$
choisir un point c (peu couvert)
| choisir une couverture ℓ couvrant c
| et l'inclure dans la solution partielle
| | supprimer les points couverts par ℓ
| | et les couvertures couvrant ces points
| | lancer la récursion sur le graphe réduit

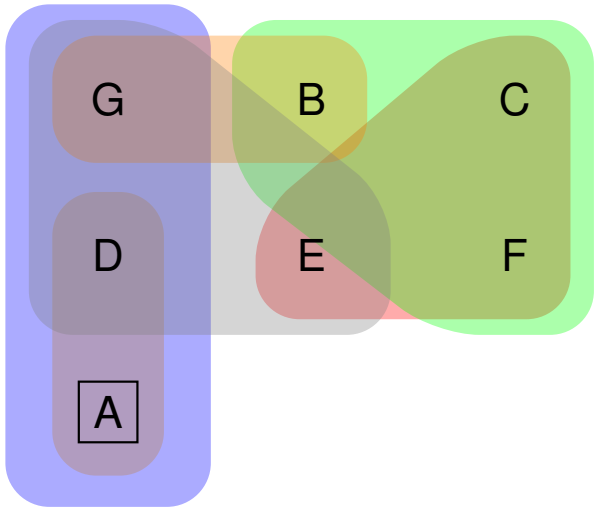
Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

A	B	C	D	E	F	G
0	0	1	0	1	1	0
1	0	0	1	0	0	1
0	1	1	0	0	1	0
1	0	0	1	0	0	0
0	1	0	0	0	0	1
0	0	0	1	1	0	1

Couverture exacte (version graphique)

Trouver une sélection de couvertures
couvrants exactement chacun des points



Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Algo X de Knuth (version graphique)

s'il n'y aucun point, renvoyer $\varnothing$
choisir un point c (peu couvert)
| choisir une couverture ℓ couvrant c
| et l'inclure dans la solution partielle
| | supprimer les points couverts par ℓ
| | et les couvertures couvrant ces points
| | lancer la récursion sur le graphe réduit

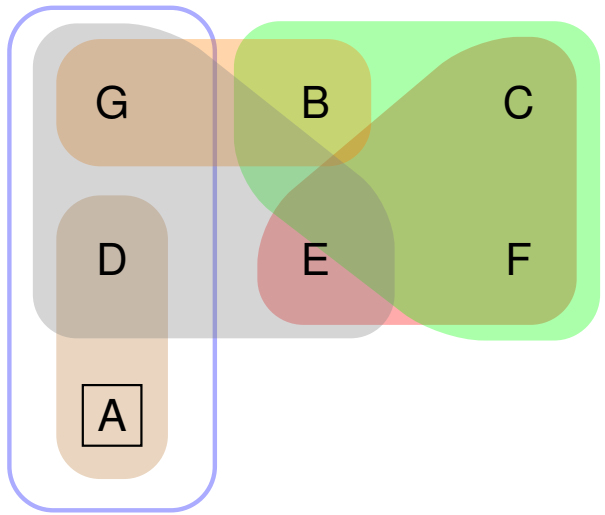
Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

A	B	C	D	E	F	G
0	0	1	0	1	1	0
1	0	0	1	0	0	1
0	1	1	0	0	1	0
1	0	0	1	0	0	0
0	1	0	0	0	0	1
0	0	0	1	1	0	1

Couverture exacte (version graphique)

Trouver une sélection de couvertures
couvrants exactement chacun des points



Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Algo X de Knuth (version graphique)

s'il n'y aucun point, renvoyer $\varnothing$
choisir un point c (peu couvert)
| choisir une couverture ℓ couvrant c
| et l'inclure dans la solution partielle
| | supprimer les points couverts par ℓ
| | et les couvertures couvrant ces points
| | lancer la récursion sur le graphe réduit

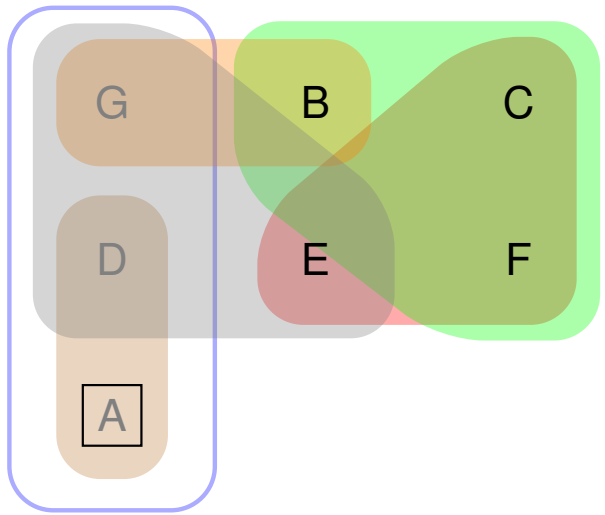
Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

A	B	C	D	E	F	G
0	0	1	0	1	1	0
1	0	0	1	0	0	1
0	1	1	0	0	1	0
1	0	0	1	0	0	0
0	1	0	0	0	0	1
0	0	0	1	1	0	1

Couverture exacte (version graphique)

Trouver une sélection de couvertures
couvrants exactement chacun des points



Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Algo X de Knuth (version graphique)

s'il n'y aucun point, renvoyer $\varnothing$
choisir un point c (peu couvert)
| choisir une couverture ℓ couvrant c
| et l'inclure dans la solution partielle
| | supprimer les points couverts par ℓ
| | et les couvertures couvrant ces points
| | lancer la récursion sur le graphe réduit

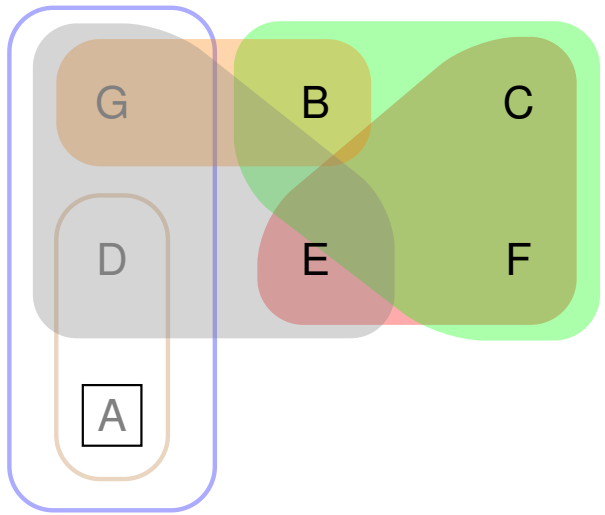
Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

A	B	C	D	E	F	G
0	0	1	0	1	1	0
1	0	0	1	0	0	1
0	1	1	0	0	1	0
1	0	0	1	0	0	0
0	1	0	0	0	0	1
0	0	0	1	1	0	1

Couverture exacte (version graphique)

Trouver une sélection de couvertures
couvrants exactement chacun des points



Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Algo X de Knuth (version graphique)

s'il n'y aucun point, renvoyer $\varnothing$
choisir un point c (peu couvert)
| choisir une couverture ℓ couvrant c
| et l'inclure dans la solution partielle
| | supprimer les points couverts par ℓ
| | et les couvertures couvrant ces points
| | lancer la récursion sur le graphe réduit

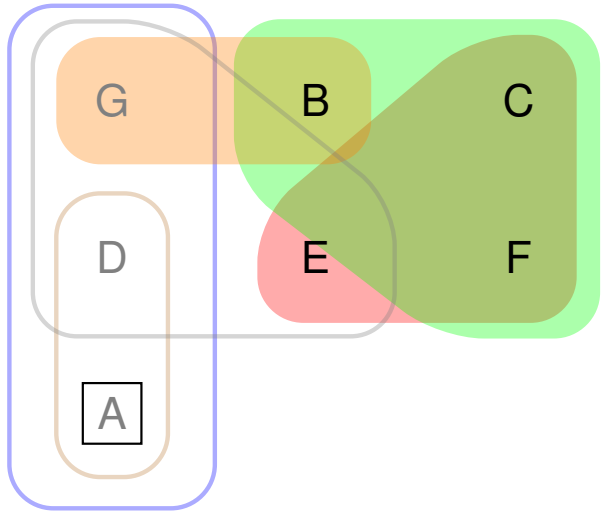
Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

A	B	C	D	E	F	G
0	0	1	0	1	1	0
1	0	0	1	0	0	1
0	1	1	0	0	1	0
1	0	0	1	0	0	0
0	1	0	0	0	0	1
0	0	0	1	1	0	1

Couverture exacte (version graphique)

Trouver une sélection de couvertures
couvrants exactement chacun des points



Algo X de Knuth (version matricielle)

si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Algo X de Knuth (version graphique)

s'il n'y aucun point, renvoyer $\varnothing$
choisir un point c (peu couvert)
| choisir une couverture ℓ couvrant c
| et l'inclure dans la solution partielle
| | supprimer les points couverts par ℓ
| | et les couvertures couvrant ces points
| | lancer la récursion sur le graphe réduit

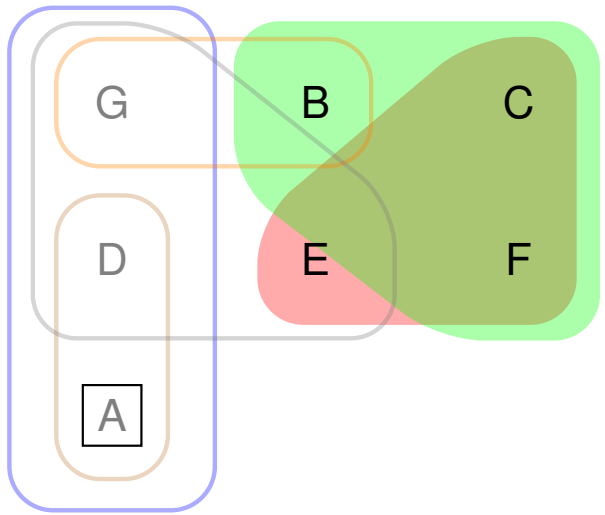
Couverture exacte (version matricielle)

Trouver une sélection de lignes
dont la somme est une ligne remplie de 1

A	B	C	D	E	F	G
0	0	1	0	1	1	0
1	0	0	1	0	0	1
0	1	1	0	0	1	0
1	0	0	1	0	0	0
0	1	0	0	0	0	1
0	0	0	1	1	0	1

Couverture exacte (version graphique)

Trouver une sélection de couvertures
couvrants exactement chacun des points

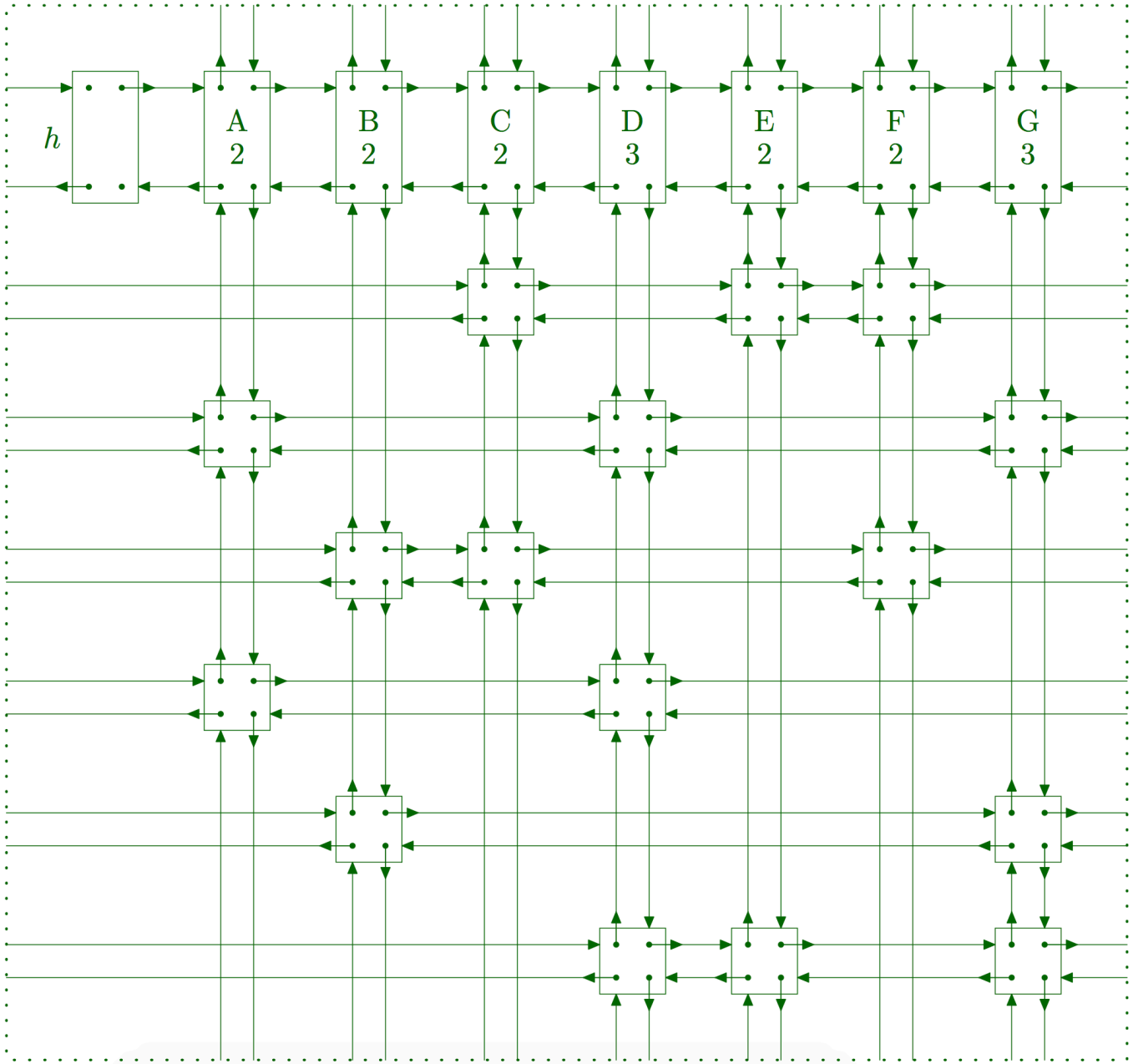


Algo X de Knuth (version matricielle)

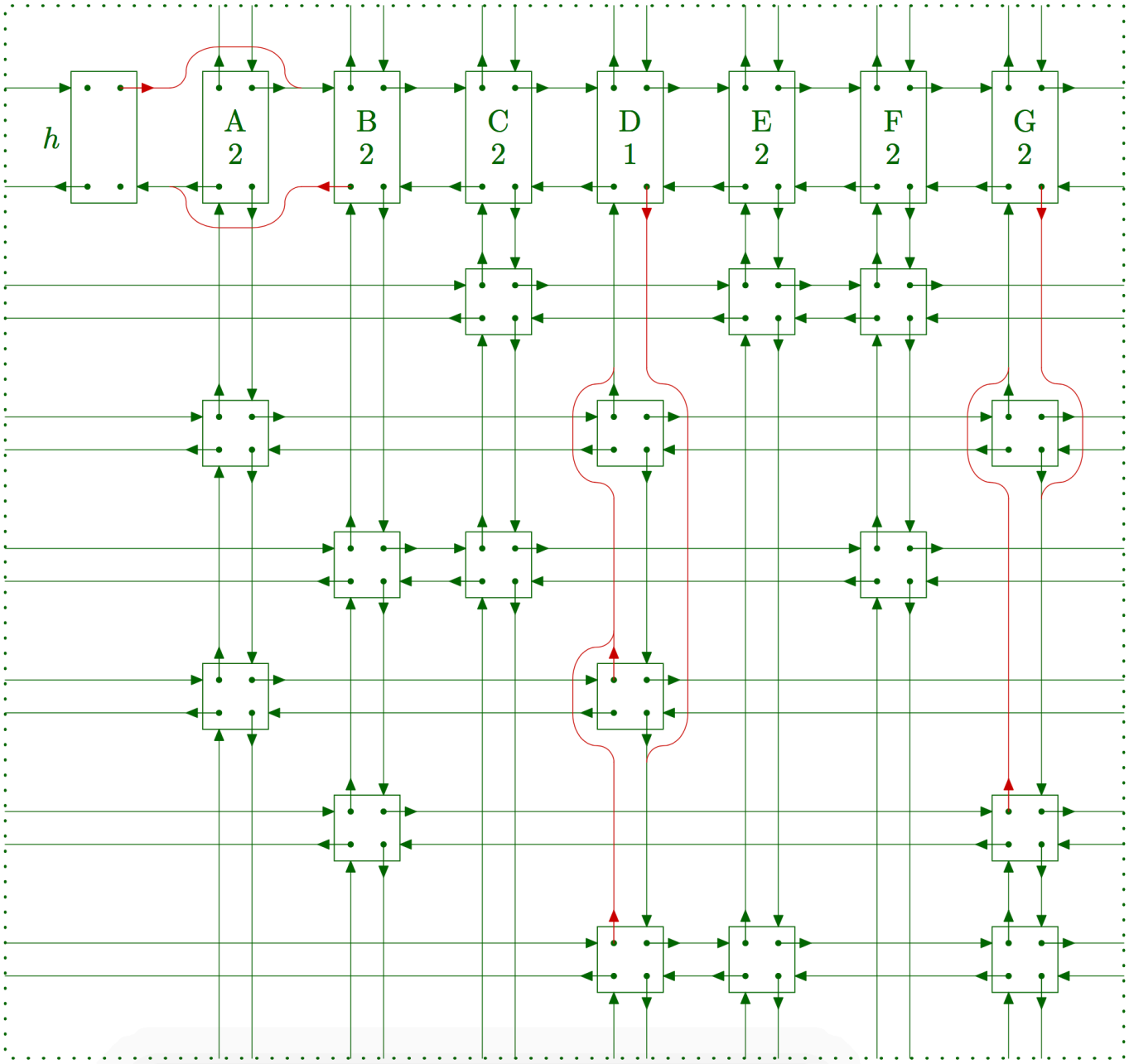
si la matrice est vide, renvoyer $\varnothing$
choisir une colonne c (ayant peu de 1)
| choisir une ligne ℓ ayant un 1 dans c
| et l'inclure dans la solution partielle
| | supprimer les colonnes ayant un 1 dans ℓ
| | et les lignes ayant un 1 dans ces colonnes
| | lancer la récursion sur la matrice réduite

Algo X de Knuth (version graphique)

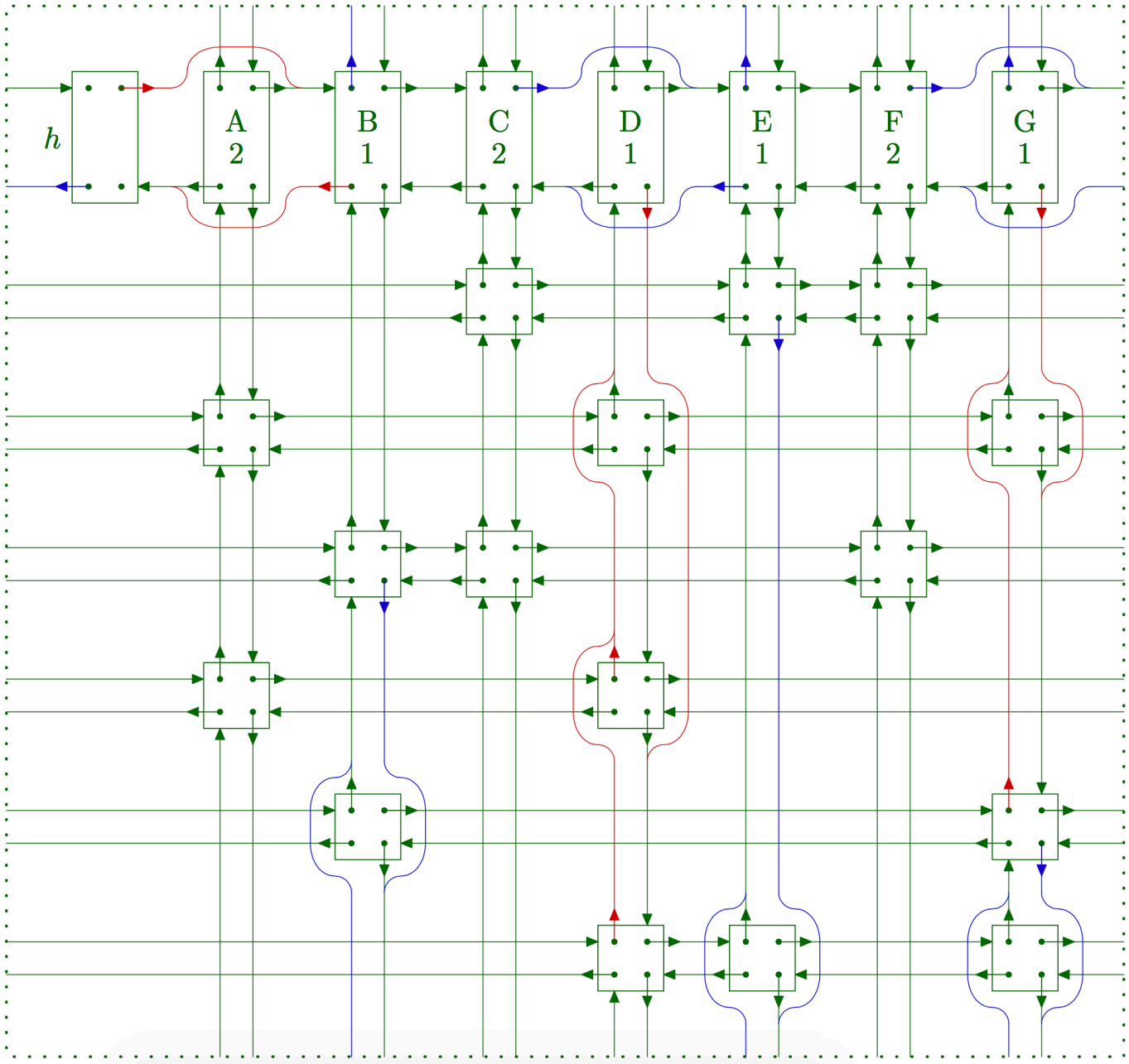
s'il n'y aucun point, renvoyer $\varnothing$
choisir un point c (peu couvert)
| choisir une couverture ℓ couvrant c
| et l'inclure dans la solution partielle
| | supprimer les points couverts par ℓ
| | et les couvertures couvrant ces points
| | lancer la récursion sur le graphe réduit



source: Donald E. Knuth https://arxiv.org/abs/cs/0011047



source: Donald E. Knuth https://arxiv.org/abs/cs/0011047



source: Donald E. Knuth https://arxiv.org/abs/cs/0011047