Problème d'énumération des parties d'un ensemble

Jean-Marc Vincent[1]

[1] Laboratoire LIG
Équipe-Projet POLARIS
Jean-Marc.Vincent@imag.fr

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

MOTIVATIONS

Algorithme de recherche

- Trouver toutes les manières de placer n reines sur un échiquier sans qu'elles soient en prise.
- Dans un sac à dos charger un ensemble des objets ayant une valeur totale maximale (contrainte de volume)
- Trouver les configurations du jeu Tic-Tac-Toe
- Trouver les k-coloriages d'un graphe
- etc

MOTIVATIONS

Algorithme de recherche

- Trouver toutes les manières de placer n reines sur un échiquier sans qu'elles soient en prise.
- Dans un sac à dos charger un ensemble des objets ayant une valeur totale maximale (contrainte de volume)
- Trouver les configurations du jeu Tic-Tac-Toe
- Trouver les k-coloriages d'un graphe
- etc

Le problème SAT

$$\Phi(x_1, x_2, , \cdots, x_n) = \bigwedge_{i=1}^{m} C_i$$

$$C_i = x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_{k_i}},$$

- n variables
- m clauses, k_i taille de la clause C_i avec avec x_{i_l} l'une des variables x_j ou sa négation.
- Forme normale conjonctive

Problème de la classe $\mathcal{NP}$

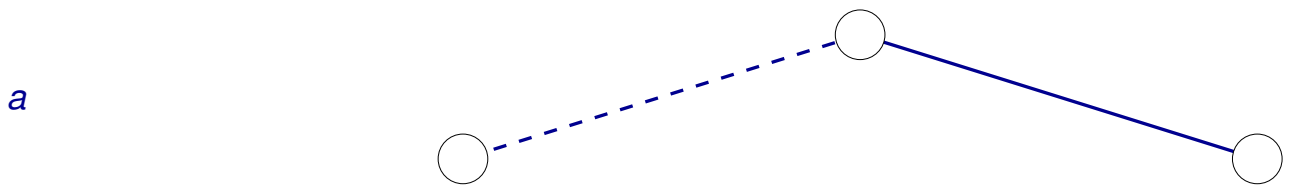
UFR IM²AG
UNIVERSITÉ Grenoble Alpes

EXEMPLE : $n = 4$

$$\mathcal{E} = \{a, b, c, d\}$$

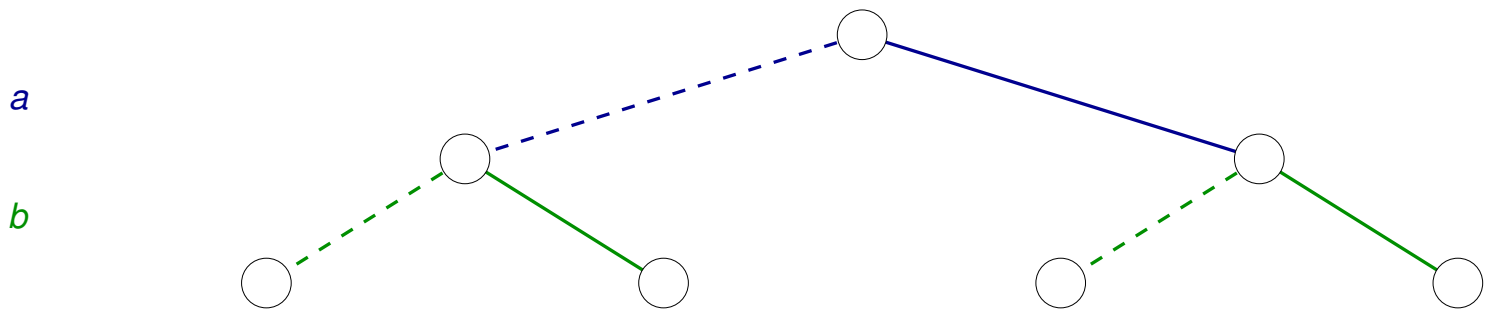
EXEMPLE : $n = 4$

$$\mathcal{E} = \{a, b, c, d\}$$



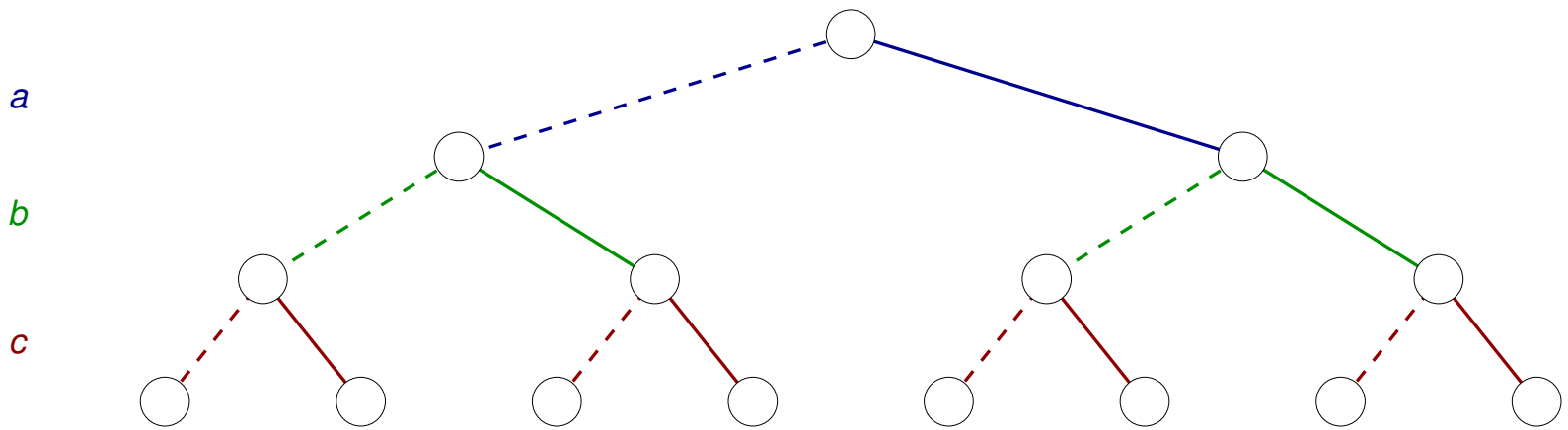
EXEMPLE : $n = 4$

$$\mathcal{E} = \{a, b, c, d\}$$



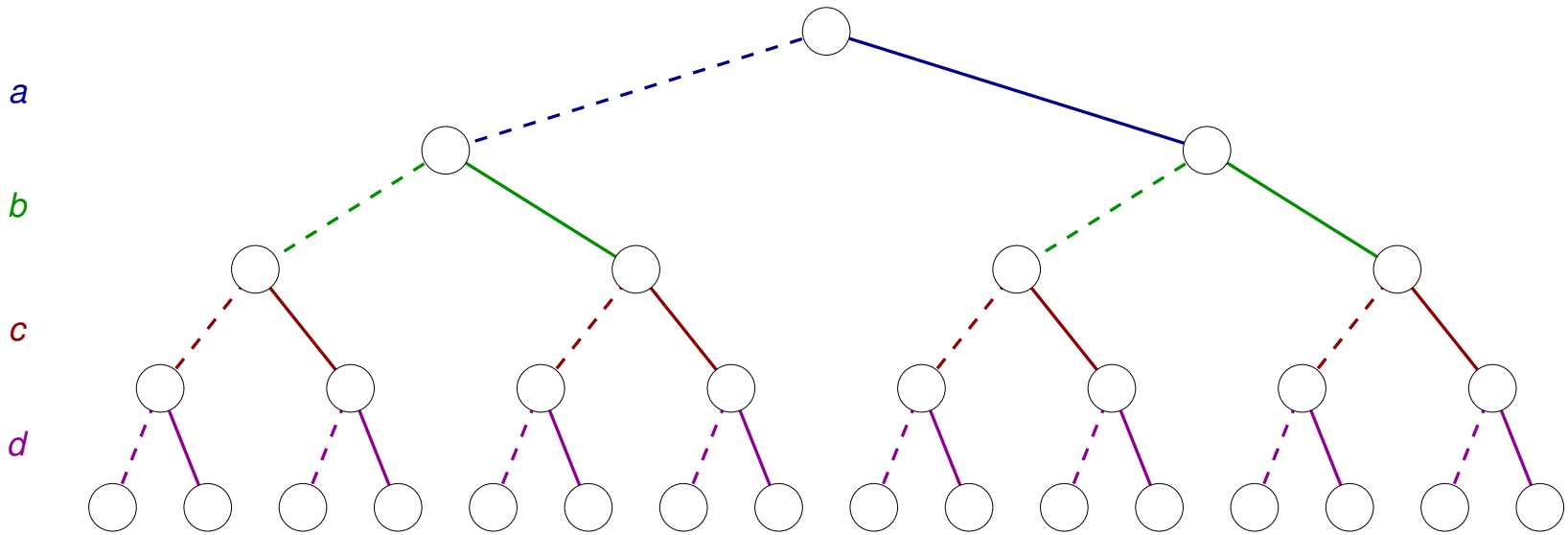
EXEMPLE : $n = 4$

$$\mathcal{E} = \{a, b, c, d\}$$



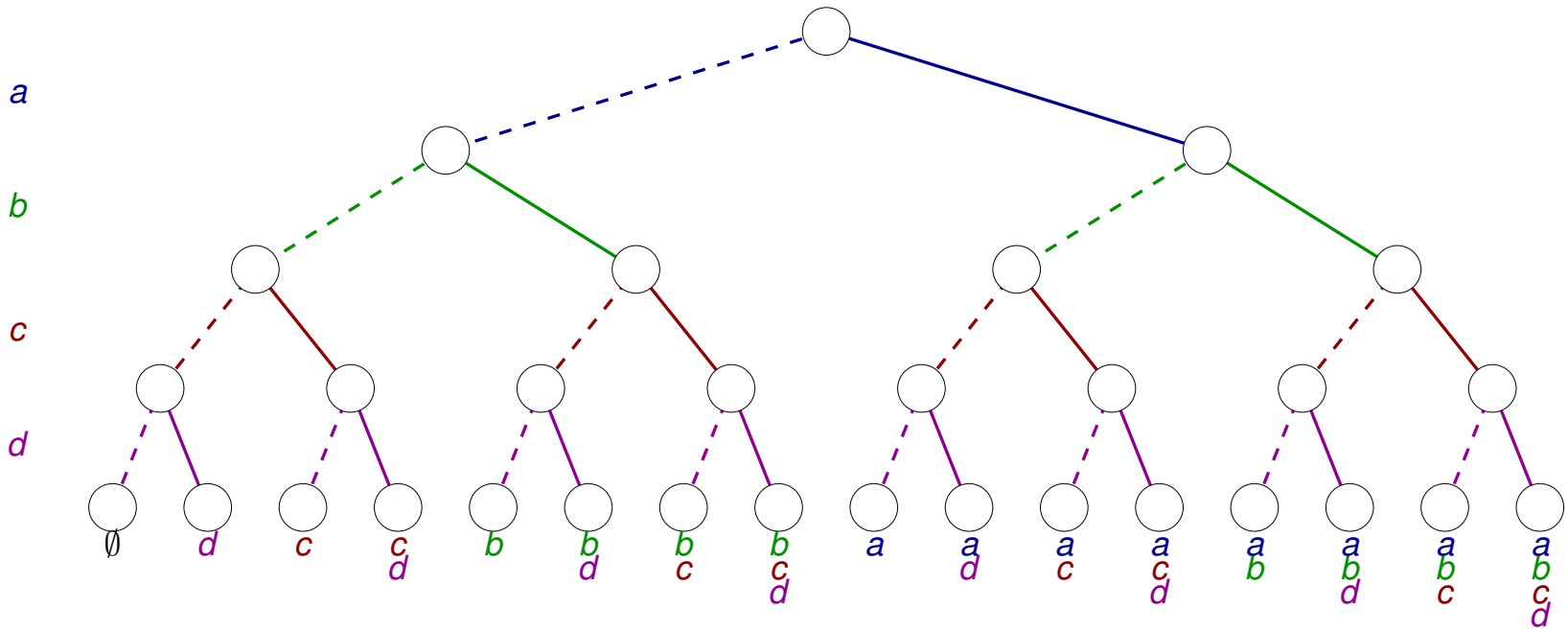
EXEMPLE : $n = 4$

$\mathcal{E} = \{a, b, c, d\}$



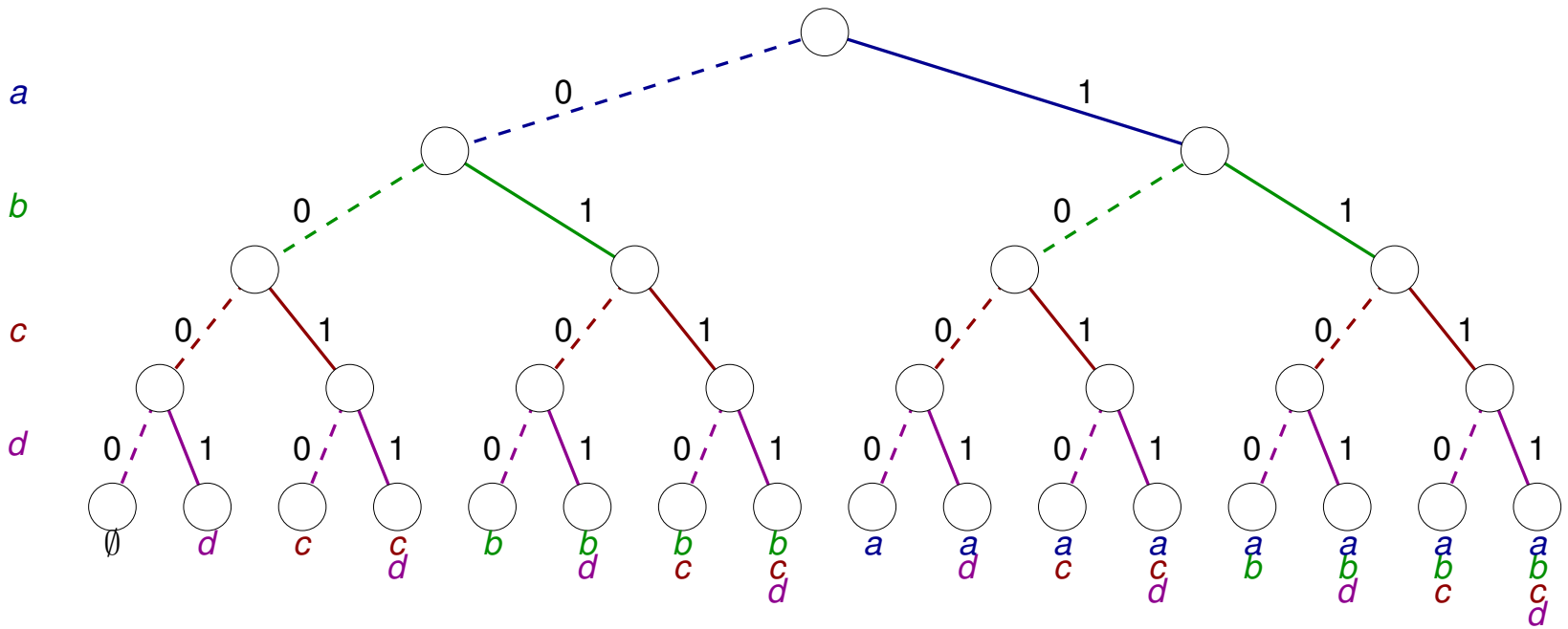
EXEMPLE : $n = 4$

$\mathcal{E} = \{a, b, c, d\}$



EXEMPLE : $n = 4$

$\mathcal{E} = \{a, b, c, d\}$



À chaque élément on associe un booléen selon qu'il est présent ou non dans la partie.
On a une **bijection** entre l'ensemble des parties d'un ensemble de taille n et l'ensemble des vecteurs de bits de dimension n (choix d'un rang par élément)

Ici la partie $\{b, d\}$ pourrait être codée par

a	b	c	d
0	1	0	1

ÉNUMÉRATION DIRECTE DES PARTIES D'UN ENSEMBLE

procédure `énumérer_Codage` (Y)
Données: $Y = \{y_0, \cdots, y_{n-1}\}$ ensembles d'éléments
Résultat: Une et une seule visite de chaque partie de Y

for $i = 0$ **to** $2^n - 1$
  Écrire i en binaire
  Convertir le vecteur de bits en une partie p_i de Y
  `Visiter` p

- Coût de l'énumération 2^n étapes
- Parcourt fixé par la numérotation des éléments de Y
- Difficile de faire le lien entre le nombre entier i et des propriétés de la partie p_i (par exemple le cardinal)

ÉNUMÉRATION RÉCURSIVE DES PARTIES D'UN ENSEMBLE

procédure `énumérer_visiter_parties` (X, Y)

if $X = \emptyset$
  `Visiter` (Y)
else
  $x =$`Choisir` (X)
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y)$
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y \cup \{x\})$

- Comment se fait le passage de paramètres ?
- Quel est l'appel initial ?
- Que fait l'algorithme ?
- Quel est le coût de cet algorithme ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

ÉNUMÉRATION RÉCURSIVE DES PARTIES D'UN ENSEMBLE

procédure `énumérer_visiter_parties` (X, Y)

if $X = \emptyset$
  `Visiter` (Y)
else
  $x =$`Choisir` (X)
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y)$
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y \cup \{x\})$

- Comment se fait le passage de paramètres ? passage par valeurs
- Quel est l'appel initial ?
- Que fait l'algorithme ?
- Quel est le coût de cet algorithme ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

ÉNUMÉRATION RÉCURSIVE DES PARTIES D'UN ENSEMBLE

procédure `énumérer_visiter_parties` (X, Y)

if $X = \emptyset$
  `Visiter` (Y)
else
  $x =$`Choisir` (X)
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y)$
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y \cup \{x\})$

- Comment se fait le passage de paramètres ? passage par valeurs
- Quel est l'appel initial ? `énumérer_visiter_parties` $(X, \emptyset)$
- Que fait l'algorithme ?
- Quel est le coût de cet algorithme ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

ÉNUMÉRATION RÉCURSIVE DES PARTIES D'UN ENSEMBLE

procédure `énumérer_visiter_parties` (X, Y)
Données: X et Y ensembles disjoints d'éléments
Résultat: Une et une seule visite de chaque partie de $X \cup Y$ de la forme $p \cup Y$ avec $p \subset X$

if $X = \emptyset$
  `Visiter` (Y)
else
  $x =$`Choisir` (X)
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y)$
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y \cup \{x\})$

- Comment se fait le passage de paramètres ? passage par valeurs
- Quel est l'appel initial ? `énumérer_visiter_parties` $(X, \emptyset)$
- Que fait l'algorithme ?
- Quel est le coût de cet algorithme ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

ÉNUMÉRATION RÉCURSIVE DES PARTIES D'UN ENSEMBLE

procédure `énumérer_visiter_parties` (X, Y)
Données: X et Y ensembles disjoints d'éléments
Résultat: Une et une seule visite de chaque partie de $X \cup Y$ de la forme $p \cup Y$ avec $p \subset X$

if $X = \emptyset$
  `Visiter` (Y)
else
  $x =$`Choisir` (X)
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y)$
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y \cup \{x\})$

- Comment se fait le passage de paramètres ? passage par valeurs
- Quel est l'appel initial ? `énumérer_visiter_parties` $(X, \emptyset)$
- Que fait l'algorithme ?
- Quel est le coût de cet algorithme ? 2^n appels, avec n la taille de X

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

ÉNUMÉRATION DES PARTIES D'UN ENSEMBLE : PREUVE

procédure `énumérer_visiter_parties` (X, Y)
Données: X et Y ensembles disjoints d'éléments
Résultat: Une et une seule visite de chaque partie de $X \cup Y$ de la forme $p \cup Y$ avec $p \subset X$

if $X = \emptyset$
  `Visiter` (Y)
else
  $x =$`Choisir` (X)
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y)$
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y \cup \{x\})$

- Que faut-il prouver ?
- Comment faire la preuve ?

ÉNUMÉRATION DES PARTIES D'UN ENSEMBLE : PREUVE

procédure `énumérer_visiter_parties` (X,Y)
Données: X et Y ensembles disjoints d'éléments
Résultat: Une et une seule visite de chaque partie de $X \cup Y$ de la forme $p \cup Y$ avec $p \subset X$

if $X = \emptyset$
  `Visiter` (Y)
else
  $x =$`Choisir` (X)
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y)$
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y \cup \{x\})$

- Que faut-il prouver ?
  `énumérer_visiter_parties` $(X,\emptyset)$ visite une et une seule fois chaque partie de X
- Comment faire la preuve ?

ÉNUMÉRATION DES PARTIES D'UN ENSEMBLE : PREUVE

procédure `énumérer_visiter_parties` (X, Y)
Données: X et Y ensembles disjoints d'éléments
Résultat: Une et une seule visite de chaque partie de $X \cup Y$ de la forme $p \cup Y$ avec $p \subset X$

if $X = \emptyset$
  `Visiter` (Y)
else
  $x =$`Choisir` (X)
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y)$
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y \cup \{x\})$

- Que faut-il prouver ?

  `énumérer_visiter_parties` $(X, \emptyset)$ visite une et une seule fois chaque partie de X
- Comment faire la preuve ?
  - Correction partielle
  - Terminaison

CORRECTION (1)

procédure `énumérer_visiter_parties` (X, Y)

if $X = \emptyset$
  `Visiter` (Y) **(0)**
else
  $x =$`Choisir` (X)
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y)$ **(1)**
  `énumérer_visiter_parties` $(X \backslash \{x\}, Y \cup \{x\})$ **(2)**

Correction partielle

- pré-condition de l'appel : X et Y sont 2 ensembles disjoints
- Si $X = \emptyset$ (on est dans le cas de base **(0)**) la récursion s'arrête et on visite Y, comme Y est la seule partie de $X \cup Y$ contenant Y (car X est vide) le résultat est correct.
- Si $X \neq \emptyset$ (on est dans le cas général) alors les paramètres des appels récursifs **(1)** et **(2)** sont également disjoints. Pour un élément x donné de X, les parties de $X \cup Y$ contenant Y, soit ne contiennent pas x et sont donc visitées par l'appel **(**1), soit contiennent x et sont donc visitées par l'appel **(2)**. Ici chaque partie visitée ne l'est qu'une seule fois car les deux appels visitent des parties différentes.
- lors de l'appel initial la pré-condition est vérifiée
- au retour de l'appel toutes les parties de $X \cup Y$ contenant Y ont été visitées

CORRECTION (2)

```
procédure énumérer_visiter_parties (X, Y)
    if X = ∅
     └ Visiter (Y)                                              (0)
    else
     │ x = Choisir (X)
     │ énumérer_visiter_parties (X\{x}, Y)                      (1)
     └ énumérer_visiter_parties (X\{x}, Y ∪ {x})                 (2)
```

Terminaison

- Soit $V(X, Y) = |X|$,
- $V(X, Y)$ est un entier positif et strictement décroissant sur l'arbre des appels.

Donc l'arbre des appels est fini et donc l'appel se termine.

CORRECTION D'UNE PROCÉDURE RÉCURSIVE

Correction partielle

- Principe de la preuve (de correction partielle) d'une procédure récursive

  Si, avec l'hypothèse que les appels récursifs sont corrects et que les paramètres vérifient la pré-condition, on prouve que le corps de la procédure est correct, alors le code de la procédure est correct.
- Appel initial

  Les paramètres de l'appel initial vérifient la pré-condition.
- Terminaison

  Si l'appel se termine alors la post-condition implique la correction de l'algorithme.

Terminaison

- Variant associé aux appels
- V fonction des paramètres d'appel (entier)
- V décroissant
- V borné inférieurement

l'appel récursif se termine.

COÛT DE L'ALGORITHME

```
procédure énumérer_visiter_parties (X, Y)
    if X = ∅
        Visiter (Y)                                          (0)
    else
        x = Choisir (X)
        énumérer_visiter_parties (X\{x}, Y)                  (1)
        énumérer_visiter_parties (X\{x}, Y ∪ {x})            (2)
```

- Nombre d'appels effectués par l'algorithme $C(N)$ (N taille de X lors de l'appel initial)
  remarque : le coût ne dépend que de la taille du premier paramètre de l'appel.
- Équation de récurrence

  $C(N) = C(N-1) + C(N-1)$;

  $C(0) = 1$

  $C(N) = 2^N$.

- Coût en temps exponentiel dans la taille de la donnée.
- Coût en mémoire : hauteur de l'arbre d'appel N

  $\mathcal{O}(N^2)$ si recopie des paramètres

  $\mathcal{O}(N)$ si passage par référence

EXERCICES

- Proposer une(des) structure(s) de donnée pour coder une partie d'un ensemble
- Réécrire l'algorithme avec cette structure de donnée
- Écrire la procédure lorsque le passage d'argument se fait par référence
- Adapter l'algorithme pour `Visiter` toutes les parties de cardinal k d'un ensemble. On pourra introduire des conditions d'explorations des branches de l'arbre (ne pas faire les appels inutiles).

PROBLÈME

Le problème SOMME

Étant donné un ensemble de n nombres entiers $\{x_1, \cdots, x_n\}$ et un entier S, existe-t'il un sous-ensemble d'indices distincts $\{i_1, \cdots, i_k\}$ tels que

$$x_{i_1} + \cdots x_{i_k} = S$$

Le cas échéant donner toutes les solutions possibles.

1. Écrire un algorithme de recherche d'une solution
2. Écrire la preuve de l'algorithme.
3. Proposer des conditions de branchement.
4. Programmer l'algorithme et évaluer le nombre d'appels effectués.
5. Quels pré-calculs peut-on faire pour améliorer l'algorithme ?

On pourra écrire un générateur de jeux de tests, qui à partir d'un germe de générateur aléatoire, fournit un ensemble de n entiers. On pourra faire varier la valeur de S pour analyser l'efficacité des conditions de branchement.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes