Vous DEVEZ comprendre cet exercice. Vous ne devez pas connaître l'infinité des résultats sur l'infinité des domaines des mathématiques, mais vous devez comprendre comment est organisée la preuve. Vous devez connaître le canvas de cette preuve et être capable de le reproduire. Une fois le canvas bien écrit, vous devez remplir les blancs. Et là c'est plus dur. Ca veut pas dire que c'est impossible et qu'il faut abandonner toute idée de le faire ca veut juste dire que c'est plus difficile parce que ce n'est pas du par coeur, il faut réfléchir.

TD3 : Ex 1

L'objectif de l'exercice est de montrer que PARTITION est plus difficile que SUBSET SUM. On va rajouter un objectif: Montrer que PARTITION est NP-Complet ; sachant que SUBSET SUM est NP-Complet.

(SUBSET SUM) $\leq$ (PARTITION)

Etape 1 : définir les problèmes sous forme de problème de décision, si ce n'est pas fait dans l'exercice.

Soit $x_1 x_2 - x_n \in \mathbb{N}$

$B \in \mathbb{N}$

$\exists ? I \subset [\![1, n]\!] / \sum_{i \in I} a_i = B ?$

Exemple

X 1 3 2 8 10

B= 12 : 1+3+8

==> I = {1, 2, 4}

Soit $y_1 y_2 \dots y_m \in \mathbb{N}$

$\exists ? J \subset [\![1, m]\!] / \sum_{i \in J} y_i = \sum_{i \notin J} y_i$

X = 1 2 7 9 10 13

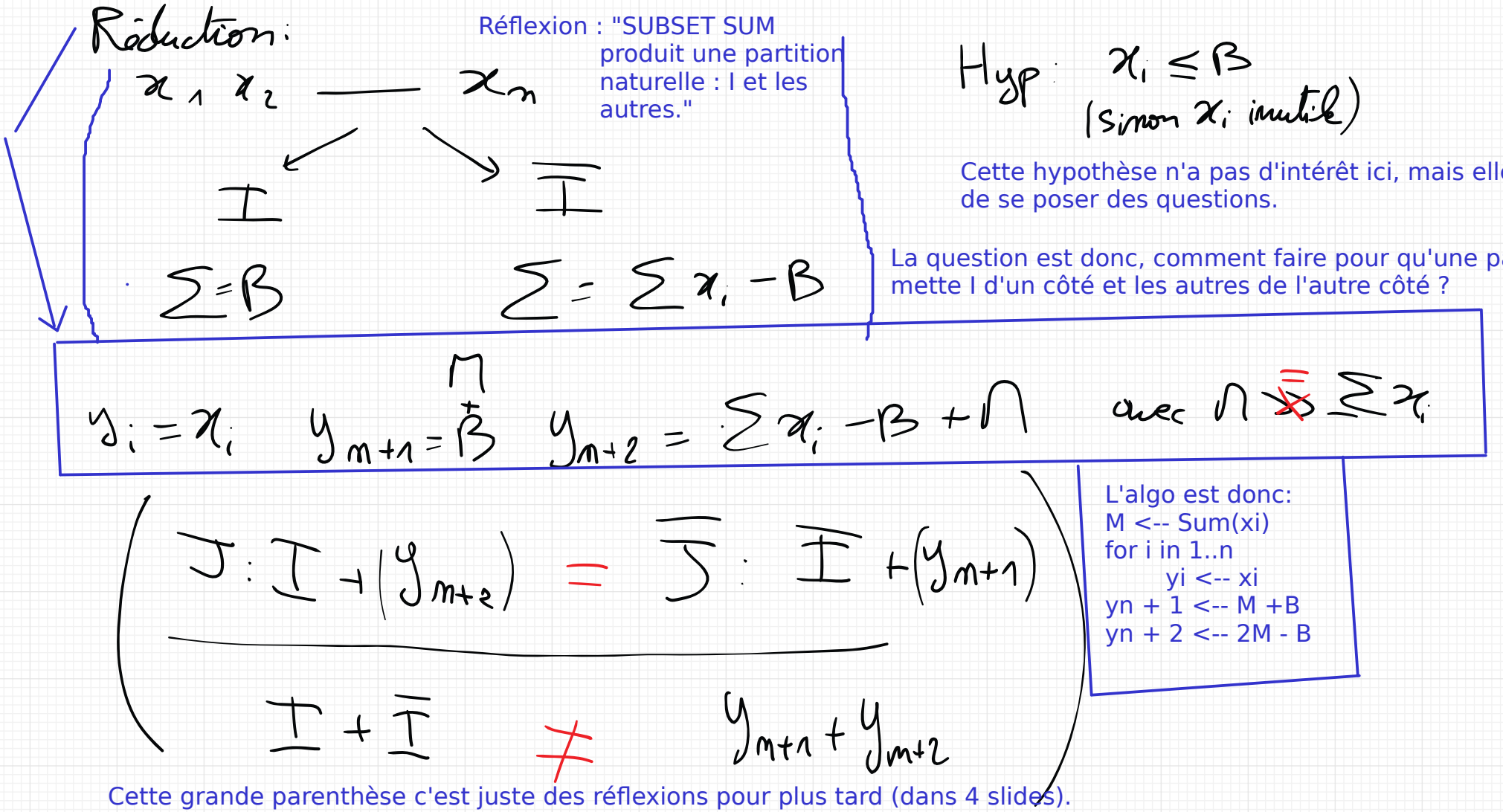
2+9+10 = 1+7+13

J = {2, 4, 5}

Etape 2 : montrer que PARTITION est dans NP. Ici on crée une machine qui 1° choisi pour tout i entre 1 et m si i est dans J ou non. puis 2°, cette machine vérifie que les deux sommes sont égales.

Complexité : O(m) (plus exactement O(m * max log y_i) pour calculer la somme).

Etape 3 : construire la réduction, c'est à dire un algo qui transforme les entrées de SUBSET SUM en PARTITION.



Cette grande parenthèse c'est juste des réflexions pour plus tard (dans 4 slides).
On vérifie ici rapidement que cette réduction ne fait pas n'importe quoi :

si je met I d'un côté et le reste de I(autre, je peux compléter avec y(n+2) à gauche et y(n+1) à droite pour égaliser les sommes.
si je fais n'imp, par exemple si je mets tous les xi à gauche, je ne peux pas égaliser les sommes.

Etape 4 : R est-il un algo polynomial vis à vis de la taille de son entrée (sinon ce n'est pas une réduction polynomiale de Karp).
Taille de l'entrée = taille de l'instance de SUBSET SUM = taille de (x1 x2, ... xn , B)

Ici on a détaillé les complexités pour les sommes et les multiplications.
Vous n'êtes pas obligés de le faire à chaque fois.
Vous pouvez considérer que les opérations sur les entiers se font en temps constant SAUF quand c'est demandé dans l'exercice.
Ici c'est important de considérer que les entiers ont des tailles non négligeables.

Taille de $x_i = \Theta(\log_2(x_i))$

Taille de $(x_1 x_2 \dots x_n, B) = \Theta\left(\sum \log_2(x_i) + \log(B)\right)$

$= \Theta\left(\max\left(\log(x_i), \log(B)\right) \times n\right)$

On note ce truc "S" (cf slide suivant) pour simplifier.

$\Rightarrow$ Temps de la réduction

$n \leftarrow \sum x_i \quad \Theta\left(n \max \log(x_i)\right) = \Theta(S)$

copie $y_i \leftarrow x_i \quad \Theta(\log(x_i)) = \Theta(S)$

$y_{n+1} \leftarrow \Pi + B \quad \Theta\left(\max(\log(\Pi), \log(B))\right) = \Theta\left(n \max(\log(x_i), \log(B))\right) = \Theta(S)$

$y_{n+2} \leftarrow 2\Pi - B \quad$ idem $= \Theta(S)$

Bien entendu, si on suppose que les copies et les additions se font en temps constant, alors la complexité de l'algo est O(n) puisqu'on fait une somme de N entiers pour calculer M, puis n + 2 copies.

$S = \max(\log(x_i)\ \log(B)) \times n$

$\longrightarrow$ complexité en temps : $\Theta((n+3)S) = \Theta(S^2)$

$\uparrow$ $O(S)$

$\longrightarrow$ polynomial en S $\longrightarrow$ polynomial en la taille de l'entrée

Ex :

$x_1\ x_2\ x_3\ x_4\ x_5 = 1\ 3\ 7\ 8\ 10$

$B = 12$

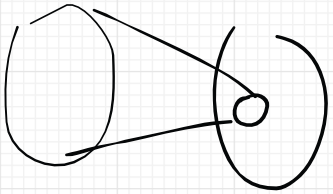
alors $n = 29$

Taille de cette entrée :
1 + 2 + 3 + 4 + 4 + 4 bits pour coder les entiers
(auquels on rajoute 5 blancs pour séparer les entiers et pouvoir les distinguer)

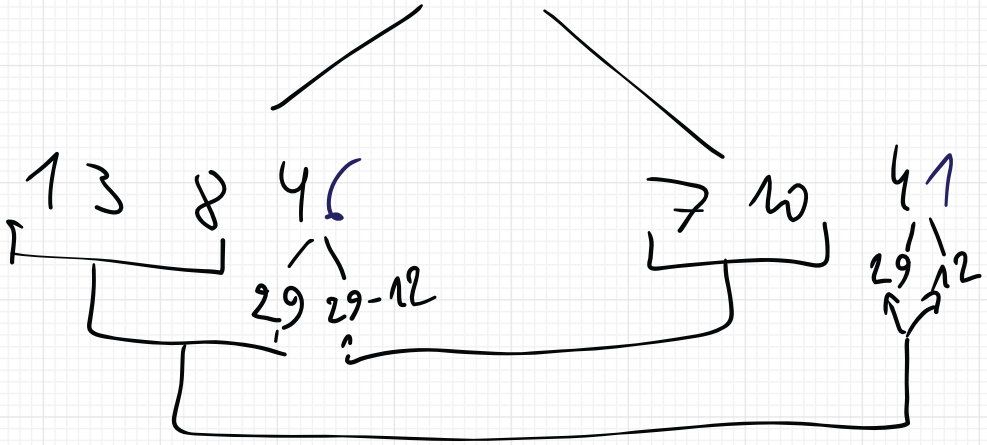
$12+29 = 41$ $\quad 2\cdot 29-12 = 46$

Instance de PARTITION sortie de R

$y_1\ y_2\ y_3\ y_4\ y_5\ y_6\ y_7 = 1\ 3\ 7\ 8\ 10$



Solution de cette instance :



On voit ici qu'un des 29 du 46 se compense avec le 29 du 41.
Le 1, 3, 8 se compensent avec le 12 du 41.
le 7 et le 10 se compensent avec le 29 - 12 du 46.

On a bien égalité entre les sommes.

Etape 5 : montrer que les instances positives (resp négatives) sont transformées en instances positives (resp négatives).
C'est l'étape la plus difficile.
Pour être exact, une moitié de l'étape est difficile. L'autre moitié est normalement facile car la réduction est construite pour ça.

Mq $(x_1 x_2 - x_n B)$ positive $\Leftrightarrow$ $(y_1 y_2 y_3 \dots y_m)$ positive.

$\Rightarrow$ $\exists I \subset [\![1,n]\!] / \sum_{i \in I} x_i = B$

Preuve au format classique :
supposons que l'instance x1 x2, ..., xn B soit positive, alors par définition de SUBSET SUM, il existe I tel que sum(xi, i in I) = B
Montrons que y1 y2 ... ym est positive ; autrement dit montrons qu'il existe J tel que
sum(yj, j in J) = sum(yj, j not in J)
Il suffit donc de trouver et de décrire ce J

Posons $J = I \cup \{n+2\}$

$$\sum_{i \in J} y_i = \sum_{i \in I} x_i + y_{n+2} = B + 2n - B = 2n$$

Ces égalités ne sortent pas de nulle part, c'est juste ce qui est écrit dans la réduction.

$$\sum_{i \notin J} y_i = \sum_{i \notin I} x_i + y_{n+1} = (n - B) + B = 2n \quad \times n$$

$\Rightarrow \exists J / \sum_{i \in J} y_i = \sum_{i \notin J} y_i \Rightarrow (y_1 y_2 \dots y_m)$ positive

Autre sens :

Preuve au format classique :
supposons que l'instance y1 y2 ... ym est positive ;
alors par définition de PARTITION, il existe J tel que
sum(yj, j in J) = sum(yj, j not in J)
Montrons que x1 x2, ..., xn B est positive ; autrement dit
montrons qu'il existe I tel que
sum(xi, i in I) = B
Il suffit donc de trouver et de décrire ce I

$\Leftarrow$ $(y_1\ y_2 \dots y_m)$ positif

$\exists J \subset [\![1, m]\!]$ / $\sum_{i \in J} y_i = \sum_{i \notin J} y_i$

Mais là, c'est un peu plus dur.

$y_{n+1} = \Pi + B \quad y_{n+2} = 2\Pi - B$ et $\sum_{i=1}^{n} y_i = \Pi$

donc si $(n+1)$ et $(n+2) \in J$

On montre d'abord que yn+1 et yn+2 ne peuvent pas être du même côté de la partition : ils ne peuvent être tous les deux dans J ou tous les duex en dehors de J

et $\left| \sum_{i \notin J} y_i \leq \sum_{i=1}^{n} y_i \leq \Pi \right.$

et $\left| \sum_{i \in J} y_i \geq y_{n+1} + y_{n+2} \geq 3\Pi \right.$

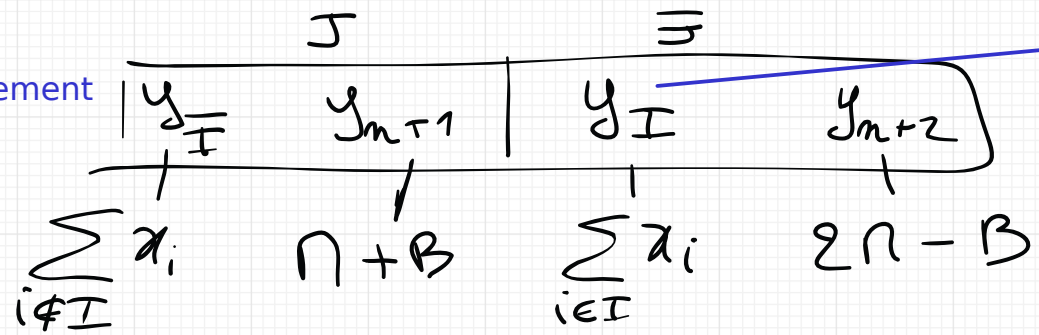
En effet, sinon les sommes des éléments de J et de non J ne peuvent être égales.

impossible

$\Rightarrow (n+1) \in J$ et $(n+2) \notin J$ (ou l'inverse)

On sait que yn+1 est dans J et yn+2 n'est pas dans J.
Rappelez vous qu'on a supposé que J existe ; donc on ne se pose pas de question, on sait que cette solution est possible c'est dans l'hypothèse. On peut donc utiliser cette solution comme si on l'avait sous les yeux. Question : où sont les éléments y1, y2 ... yn dans cette solution ? Une partie est dans J et une autre ne l'est pas. On appelle I l'ensemble des indices i tels que yi n'est pas dans J.

schématiquement
Ca donne ça



← la solution est donc sous cette forme

donc $\exists I / \sum x_i = B$
↳ $(x_1, x_2 - x_n B)$ positive

Ok, on a réussi à construire un ensemble I
Il reste plus qu'à prouver qu'il s'agit d'une solution de SUBSET SUM.

$$\text{My} \sum_{i \in I} x_i = B$$

$$\cancel{2}B = \cancel{2} \sum_{i \in I} x_i$$

on sait que $\sum_{i \in J} y_i = \sum_{i \in \bar{J}} y_i$ parce que J est une solution.

$$n - \sum_{i \in I} x_i \leftarrow \sum_{i \in \bar{I}} x_i + n + B = \sum_{i \in I} x_i + 2n - B$$

par définition de M

$$\cancel{n} - \sum_{i \in I} x_i + \cancel{n} + B = \sum_{i \in I} x_i + \cancel{2}\cancel{n} - B$$

Etape 6 : conclure que R est une réduction polynomiale de Karp et que SUBSET SUM <= PARTITION.

- Réduction en temps poly
- $(x_1 x_2 \dots x_n B)$ positif $\Leftrightarrow$ $(y_1 y_2 \dots y_m)$ positive

On a donc une réduction polynomiale de Karp de

(SUBSET SUM) vers (PARTITION)

$\Rightarrow$ (SUBSET SUM) $\leq$ (PARTITION)

---

or (SUBSET SUM) NP-Complet $\Rightarrow$ (PARTITION) NP-Difficile

or (PARTITION) $\in$ NP $\Rightarrow$ (PARTITION) NP-Complet.

Etape 7 : conclure que PARTITION est NP-Difficile
Etape 8 : conclure que PARTITION est NP-Complet.

Allez, une réduction bonus : on le fait dans l'autre sens.

On oublie que SUBSET est NP-Compelt et on oublie la réduction précédente.
Supposons que PARTITION est NP-Complet, montrons que SUBSET SUM est NP-Complet.

On suppose dans la suite que les instances de PARTITION vérifie : sum(yi) est pair. Sinon l'instance n'a trivialement pas de solution.

Etape 1 : la même qu'ici.
Etape 2 : SUBSET SUM est dans NP, en effet, la machine suivant résout le problème:
Certificat : pour tout i de 1 à n, décider si i est dans I ou non.
Vérifieur : est-ce que sum(xi, i in I) = B ?
Cette machine se calcule en temps O(n) (ou O(n max(log xi, log B))) si on veut être précis.

Etape 3 : construire R.
Soit y1 y2 ... ym une instance de PARTITION.
B <-- sum(yi) / 2
xi <-- yi
(et on a n <-- m)

Etape 4 : complexité de R,
L'entrée a une taille O(sum(log(yi))) = O(m max log yi) Posons S = m max log(yi)
Le calcul de B se fait en temps O(m max log yi) = O(S)
Le calcul de xi se fait en temps O(log(yi))
Le calcul de tous les xi se fait en temps O(sum log(yi)) = O(S)
Donc R a une complexité O(S)

Etape 5 : equivalence des positivités des instances
Supposons que y1 y2 ... ym est positive
Il existe J tel que sum(yj, j in J) = sum(yj, j not in J)
Posons I = J
On sait que sum(yj) = sum(yj, j in J) + sum(yj, j not in J)
Donc sum(yj) = sum(yj, j in J) * 2
or B = sum(yj) / 2 et sum(yj, j in J) = sum(xj, j in J)
Donc B = sum(xj, j in J)
Donc B = sum(xi, i in I)
Donc x1 x2 ... xm B est positive.

Supposons que x1, x2 ... xm B est positive.
Donc il existe I tel que sum(xi, i in I) = B

Posons J = I
Alors sum(xj, j in J) = sum(xi, i in I) = B = sum(xj) / 2
Alors sum(xj, j in J) = sum(xj) - sum(xj, j in J)
Alors sum(xj, j in J) = sum(xj, j not in J)

Donc y1, y2 ... ym est positive.

Etape 6 : donc R est une réduction polynomiale de Karp et (PARTITION) <= (SUBSET SUM)
Etape 7 : or PARTITION est NP-Complet, donc SUBSET SUM est NP-Difficile
Etape 8 : or SUBSET SUM est dans NP, donc SUBSET SUM est NP-Complet.