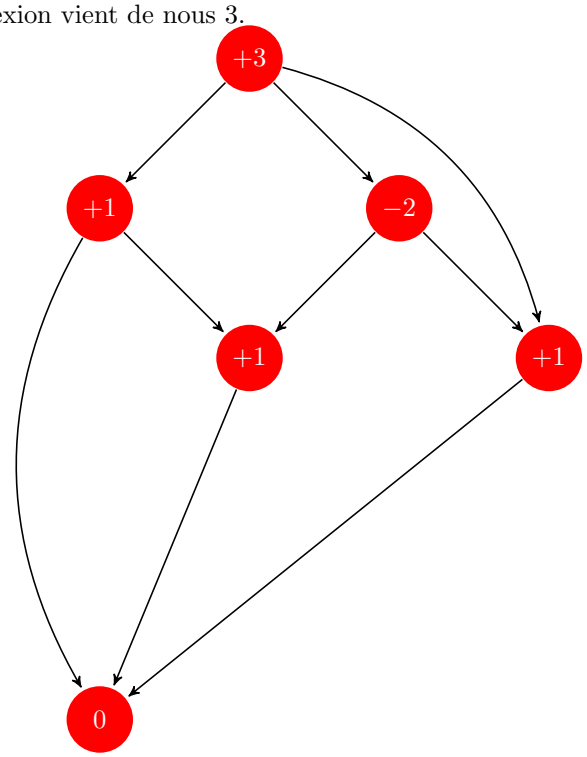
# TP3 ACT

Matthieu Caron et Armand Bour

vendredi 9 octobre 2015

**Question 1** Le code du graphique tikz est donné par Antoine Amara, mais la reflexion vient de nous 3.




**Question 2** S'ils sont tous positifs : $res = -1 * min(successeurs) - 1$
Sinon : $res = -1 * max(desValeursNegatives) + 1$

**Question 3** La fonction récursive $naif$ calcule toutes les sous-tablettes possibles. Quatre boucles sont utilisées, une pour chaque possibilité de cassure (x colonne(s) à gauche ou à droite, et x ligne(s) en haut ou en bas). Des possibilités sont alors calculées plusieurs fois.
En Python, la fonction prends en temps respectivement pour les configurations suivantes :

**(10, 7, 7, 3)** 4 minutes 47 secondes ;

**(10, 7, 5, 3)** 10 minutes 24 secondes.

La différence de temps est due à la “profondeur” de la tête de mort dans la tablette. Dans le second cas, elle est plus proche du centre, ce qui décuple le nombre de sous-arbres possiblites.
La complexité est exponentielle.

**Question 4** Notre algorithme est écrit en Python, et on ne sait pas pourquoi la version dynamique prend beaucoup de temps, avec des print on a compté, l'initialisation du tableau pour dynamic(100,100,50,50) prend une dizaine de secondes.

Donc dynamic(100,100,50,50) = -198.0 et dynamic(100,100,48,52) =

La version dynamique reste beaucoup plus rapide que la naive :

**(10, 7, 7, 3)** en naïf prend 4 minutes 47 secondes ;

**(10, 7, 7, 3)** en dynamique prend 0,120 secondes.

**Question 5** Nous avons écrit la fonction pour déterminer les valeurs, mais étant donné la durée de l'algorithme, nous n'avons pas pu les obtenir.

**Question 6** Tout d'abord, on remplit le tableau qui dépend de $m$,$n$,$i$ et $j$ ce qui nous fait du $O(m * n * i * j)$ En suite dans l'algo on veut faire tous les découpages possible : ceux dans la largeur $m$, et ceux dans la longueur $n$. Dans le calcul de la découpe, si le résultat a déjà été calculé, il s'agit simplemnet d'un accès au tableau, de complexité supposée en $O(1)$ (supposée car il n'y a aucune information concernant la complexité de l'objet utilisé *ndarray* en Python). Sinon, il s'agit d'un calcul de toutes les sous-tablettes de chocolat.

$$\begin{cases} \text{si deja calc alors } c(m,n) = 1 \\ \text{sinon } \sum_{k=1}^{m} c(m-k,n) + \sum_{l=1}^{n} c(m,n-l) \end{cases}$$

La complexité est alors en $O(m^2 * n^2)$.

**Question 7** Toutes ces configurations ont la même valeur car il s'agit de la même plaque de chocolat soit pivotée d'un quart de tour, un demi-tour, trois quarts de tours, ou vue de dos (symétrique).

**Question 8** L'idée est d'ajouter quand c'est possible 8 valeurs au tableau plutôt qu'une seule (la valeur habituelle et les 7 autres configurations identiques). En pratique, voici les tests effectués :

**(30, 30, 10, 8)** en dynamique classique prend 3,91 secondes;

**(30, 30, 10, 8)** en dynamique avec symétrie prend 0,78 seconde.