Un problème NP-complet ne peut a priori pas être résolu en complexité polynomiale. On peut chercher des alternatives :

- Un algorithme d'approximation qui renvoie une solution approchée en complexité polynomiale.
- Un algorithme exact qui accélère une méthode de recherche exhaustive (tout en restant en complexité exponentielle), comme la méthode Branch-and-Bound.

I Problème d'optimisation

Définition : Problème d'optimisation ♡

Un problème d'optimisation est un triplet (I, I^+, f) avec I un ensemble d'instances, $I^+ \subset I$ un ensemble de solutions et f une fonction objectif à optimiser (minimiser ou maximiser).
Un algorithme A résout ce problème d'optimisation si pour toute instance $i \in I$, $A(i)$ renvoie une solution $s \in I^+$ telle que $f(s)$ est optimum.

Exemple :

`STABLE` (optimisation)

Instance : un graphe G.
Solution : un ensemble stable de G (ensemble de sommets sans arête entre eux).
Maximiser : la taille de l'ensemble stable.

Méthode ♡ : Un problème d'optimisation peut être transformé en un problème de décision avec un seuil k. Exemple :

`STABLE` (décision)

Instance : un graphe G et un entier k.
Question : G contient-il un ensemble stable de taille $\geq k$?

Dans le cas d'une minimisation, on remplace « $\geq k$ » par « $\leq k$ » dans la question.

II Algorithme d'approximation

Définition : Algorithme d'approximation ♡

Soit $\Pi = (I, I^+, f)$ un problème d'optimisation et $\alpha \in \mathbb{R}^+$.
On dit qu'un algorithme A est une α-approximation de Π si pour toute instance x de Π :

- $A(x)$ termine et renvoie une solution $s \in I^+$.
- si Π est un problème de maximisation : $f(s) \geq \alpha f(s^*)$
- si Π est un problème de minimisation : $f(s) \leq \alpha f(s^*)$

où s^* est une solution optimale de Π.

Remarques :

- α doit être une constante indépendante de l'instance.
- $\alpha \geq 1$ (resp. ≤ 1) si Π est un problème de minimisation (resp. maximisation). Par exemple, une 2-approximation d'un problème de minimisation doit renvoyer une solution dont la valeur est au plus deux fois plus grande que la solution optimale.

Exercice 1.
Soit $G = (S, A)$ un graphe non orienté. Une couverture par sommets de G est un ensemble $S' \subset S$ tel que pour toute arête $\{u, v\} \in A$, $u \in S'$ ou $v \in S'$.

`COUVERTURE`

Instance : un graphe G et un entier k.
Question : G contient-il une couverture par sommets de taille $\leq k$?

1. On admet que `STABLE` est NP-complet. Montrer que `COUVERTURE` est NP-complet.

On appelle `COUVERTURE-OPT` le problème d'optimisation associé à `COUVERTURE` :

COUVERTURE-OPT

Instance : $G = (S, A)$ graphe.
Solution : couverture par sommet S' de G.
Minimiser : $|S'|$.

2. Montrer que l'algorithme suivant renvoie bien une couverture par sommets mais n'est pas une α-approximation de `COUVERTURE-OPT`, quel que soit α :

$S' \leftarrow \emptyset$
Tant que $A \neq \emptyset$ **:**
  Choisir $\{u, v\} \in A$
  $S' \leftarrow S' \cup \{u\}$
  Retirer de A toutes les arêtes incidentes à u
Renvoyer S'

3. Montrer que l'algorithme suivant est une 2-approximation de `COUVERTURE-OPT` :

$S' \leftarrow \emptyset$
Tant que $A \neq \emptyset$ **:**
  Choisir $\{u, v\} \in A$
  $S' \leftarrow S' \cup \{u, v\}$
  Retirer de A toutes les arêtes incidentes à u ou v
Renvoyer S'

4. Avec quelle complexité peut-on implémenter cet algorithme ?

Exercice 2.
On considère le problème :

`COUPLAGE-POIDS-MAX`

Instance : un graphe $G = (S, A)$ pondéré par $p : A \longrightarrow \mathbb{R}^+$.
Solution : un couplage C de G.
Optimisation : maximiser $p(C) = \sum_{\{u,v\} \in C} p(\{u, v\})$.

Montrer que l'algorithme suivant est une 2-approximation de `COUPLAGE-POIDS-MAX` :

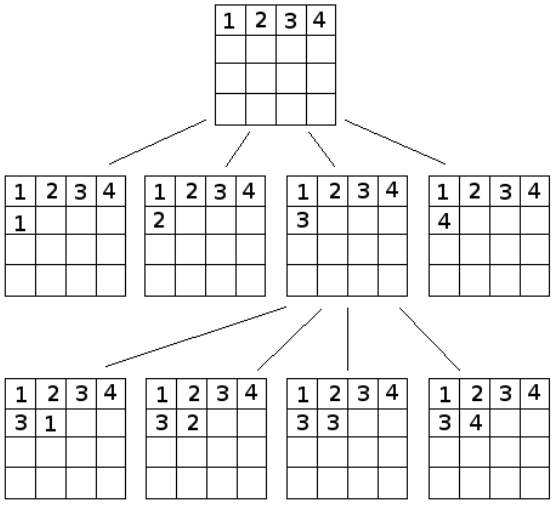
$C \leftarrow \emptyset$
Pour $a \in A$ par ordre décroissant de poids **:**
  Si $C \cup \{a\}$ est un couplage **:**
    $C \leftarrow C \cup \{a\}$
Renvoyer C

III Branch-and-Bound

Définition : Backtracking

Un algorithme par backtracking (retour sur trace) construit une solution petit à petit, en revenant en arrière s'il n'est pas possible de l'étendre en une solution complète.

Un backtracking revient à faire un parcours en profondeur sur l'arbre des solutions partielles, en passant à une branche suivante lorsqu'il n'est pas possible d'étendre une solution partielle :



Exemple : L'algorithme de Quine résout `SAT` en trouvant une valuation satisfaisant une formule logique φ par backtracking :

1. Choisir une variable x restante dans φ.

2. Tester récursivement si $\varphi[x \leftarrow 1]$ est satisfiable.
3. Si non, tester récursivement si $\varphi[x \leftarrow 0]$ est satisfiable.

L'algorithme simplifie la formule à chaque assignation de variable, ce qui peut permettre de couper des branches (si la formule est insatisfiable). Cet algorithme est donc souvent plus efficace que la recherche exhaustive, même s'il reste en complexité exponentielle dans le pire cas.

Définition : Branch-and-Bound

Un algorithme par Branch-and-Bound (séparation et évaluation) permet d'accélérer la résolution par backtracking d'un problème de maximisation $\Pi = (I, I^+, f)$ en utilisant une heuristique h telle que si x est une solution partielle qui peut être étendue en une solution $y \in I^+$ alors $f(y) \leq h(x)$.
L'algorithme conserve la meilleure solution trouvée s^* et explore un sous-arbre x que si $h(x) \geq f(s^*)$.

Remarques :

- Autrement dit : $h(x)$ majore la valeur d'une solution obtenue en étendant x. Si $h(x) < f(s^*)$, alors on ne peut pas obtenir une solution meilleure que s^* en explorant x.
- Le principe est le même pour un problème de minimisation en inversant les inégalités.
- On obtient une solution exacte (contrairement à un algorithme d'approximation) mais la complexité peut être exponentielle.

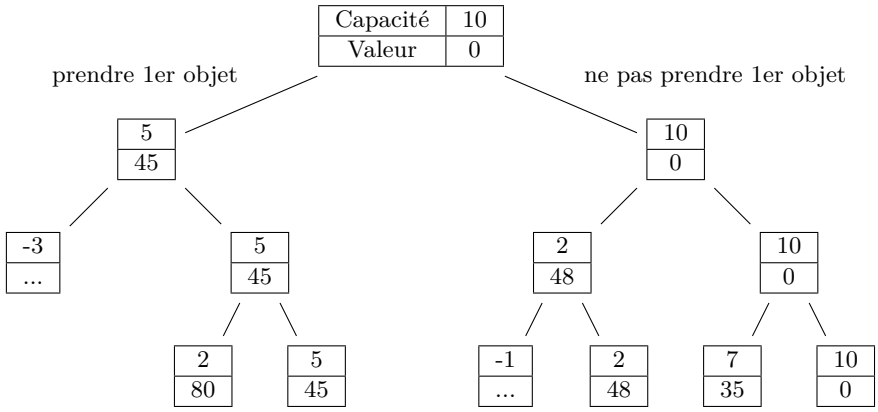
III.1 Exemple : Sac à dos

`SAC-A-DOS`

Instance : un ensemble d'objets de poids $p_1, \ldots, p_n$ et de valeurs $v_1, \ldots, v_n$ et une capacité P.
Solution : un sous-ensemble d'objets de poids total inférieur à P.
Optimisation : maximiser la valeur totale des objets.

Arbre des solutions partielles pour l'instance suivante de `SAC-A-DOS` :

Poids	5	8	3
Valeur	45	48	35



Exercice 3.

1. Donner une heuristique possible pour le problème du sac à dos.
2. Indiquer sur l'arbre ci-dessus les branches qui ne seront pas explorées par l'algorithme de Branch-and-Bound.