Rapport TP3 ACT

Gaspar Henniaux - Marwane Ouaret

1. Qu’est-ce qu’une propriété NP ?

Question 1def sum_to_partition(n ,c ,l):

```
"""
Reduce the sum problem to the partition problem

Parameters
----------
n : int
    The number of items
l : list
    The list of integers
c : int
    The sum to reach
----------
Returns
-------
n : int
    The number of items
l : list
    The list of integers
-------
"""
somme = sum(l)

if somme < c:
    raise ValueError("La cible est trop élevée pour être atteinte avec
cet ensemble.")

rep = l + [somme - 2 * c]

return rep, n+1
```

PROF

Ici, un certificat est une association pour chaque objet à un sac.

On peut utiliser un dictionnaire dont les clés seront les objets et les valeurs les sacs.

Par conséquent, la taille d'un certificat est n, le nombre d'objet. Cette taille est bien bornée polynomialement par rapport à la taille de l'entrée car n est la taille de l'entrée.

```
fonction verif_sac(certificat, n, poids, c, k):
```

```
    if len(certificat) != n:
        retourner faux

    somme : dictionnaire
    pour chaque objet dans certificat:
        si certificat[objet] n'est pas dans somme:
            somme[certificat[objet]] = poids[objet]
        sinon:
            somme[certificat[objet]] += poids[objet]

    if len(somme) != k:
        retourner faux

    pour chaque sac dans somme:
        si somme[sac] > c:
            retourner faux

    retourner vrai
```

On passe n fois dans la boucle pour remplir le dictionnaire somme, et k fois pour vérifier que chaque sac ne dépasse pas la capacité c. La complexité de cette fonction est donc en O(n + k).

Question 2

2.1.

```
fonction generer_certificat(n, k):
    certificat : dictionnaire

    pour i allant de 1 à n:
        certificat[i] = random(1, k)

    retourner certificat
```

PROF

cet algorithme génère les certificats de manière uniforme car chaque objet est associé à un sac de manière aléatoire. Chaque certificat a donc la même probabilité d'être généré.

2.2.

```
certificat = generer_certificat(n, k)
verif_sac(certificat, n, poids, c, k)
```

Question 3

3.1.

Pour n et k fixés, le nombre de certificats possibles est k^n. En effet, pour chaque objet, on a k choix de sacs possibles.

3.2.

pour ordonner les certificats, on peut les trier par ordre lexicographique.

3.3.

Pour tester si le problème a une solution ou non, on peut tester tous les certificats possibles. Si un certificat est valide, alors le problème a une solution.

La complexité de cet algorithme est en O(k^n * (n + k)). En effet, on teste tous les certificats possibles, et pour chaque certificat, on vérifie s'il est valide en O(n + k).

Question 4

voir algo

2. Réduction polynomiale

Question 1

1.

ecrire feuille

1.1.

```
function reduction (nb_objet, liste_objet)
    capacite_sac = somme de liste_objet divisé par 2
    nombre_sac = 2
    return nb_objet, liste_objet, capacite_sac, nombre_sac
```

PROF

1.2.

On a déjà prouvé que binPack est un problème NP en montrant qu'il existe un algorithme polynomial pour vérifier si un certificat est valide. On a aussi montré que que partition se réduit polynomialement à binPack en montrant qu'on peut transformer une instance de partition en une instance de binPack en temps polynomial. Etant donné que partition est NP-complet, il est également NP-dur, c'est à dire que tout problème NP se réduit polynomialement à partition. Par transitivité, tout problème NP se réduit polynomialement à binPack. binPack est donc NP-dur et NP. Il est donc NP-complet.

1.3

Nous ne pensons pas que BinPack se réduise polynomialement dans Partition car toutes instances de BinPack ne permettent pas d'avoir une instance de partition du au nombre de sac fixé à 2 et la capaité du sac fixé aussi. Uniquement certains cas de Binpack permet une instance de Partition.

Question 2

à expliquer

Question 3

voir algo et expliquer

Question 4

on emboite les réduction, voir test algo et expliquer

Question 5

on doit ajouter des objets de la maniere suivante :

on selectionne le sac de capacité max et on construit un tableau en soustrayant la capacité max avec la capacité de tous les autres sac et on ajoute cette liste d'objets aux autres.

ensuite on definit la capacité max comme capacité max de la liste de l'instance de base et le nombre de sac ne change pas