Goodies en pagaille !

Master Informatique

Arnaud Malapert,

4 novembre 2020

Université Côte d'Azur, CNRS, I3S, France
firstname.lastname@unice.fr

Problème F : Goodies en Pagailles !

L'énoncé du problème donné à la coding battle 2020 est disponible en francais et en anglais :
`https://github.com/INSAlgo/coding-battle2020`.

En résumé

1. On dispose d'un ensemble d'objets numérotés par $I = [1, n]$ avec un poids w_i et d'un sac-à-dos de capacité W.
2. Est-il possible de remplir complètement le sac-à-dos en empêchant votre Ami Bob de faire de même avec les objets restants ?

Vous allez donc résoudre des problèmes de sac-à-dos pour déterminer si vous pouvez gagner un jeu contre un adversaire.

Méta-Algorithme

Soit un oracle énumératif `solve(`I`,` W`)` qui :

- prend en argument un ensemble d'objets I et la capacité W, et
- renvoie comme résultat l'ensemble $\mathcal{S}$ de toutes les solutions.

Une solution est un sous-ensemble d'objets remplissant complétement le sac à dos de capacité W.

Question

Écrire un méta-algorithme basé sur cet oracle qui répond au problème.

Méta-Algorithme

Soit un oracle énumératif `solve(`I`,` W`)` qui :

- prend en argument un ensemble d'objets I et la capacité W, et
- renvoie comme résultat l'ensemble $\mathcal{S}$ de toutes les solutions.

Une solution est un sous-ensemble d'objets remplissant complétement le sac à dos de capacité W.

Question

Écrire un méta-algorithme basé sur cet oracle qui répond au problème.

Algorithm: Meta Merch

```
1 S ← solve(I, W)
2 forall J ∈ S do
3     if solve(I − J, W) = ∅ then
4         return TRUE
5 return FALSE
```

Comment détecter un bug ?

Cas I : une solution est invalide.
Facile : le certificat est polynomial.

On écrit un code indépendant validant une solution.

Cas II : il manque une ou plusieurs solutions.
Difficile : on ne connaît pas le nombre de solutions dans le cas général.

On écrit des tests de validation.

Comment détecter un bug ?

Exemple 3

```
10
11
6 1 1 1 1 2 1 7 5 2 1
```

- Vous gagnez uniquement si vous prenez les objets $1+1+1+1+1+1+2+2=10$.
- Sinon, Bob pourra lui aussi tout dépenser. Par ex. si vous prenez $5, 2, 2, 1$.

Comment détecter un bug ?

Exemple 3

```
10
11
6 1 1 1 1 2 1 7 5 2 1
```

- Vous gagnez uniquement si vous prenez les objets $1+1+1+1+1+1+2+2=10$.
- Sinon, Bob pourra lui aussi tout dépenser. Par ex. si vous prenez $5, 2, 2, 1$.

Exemple 4

```
9
9
2 2 2 2 3 3 3 4 4
```

- Vous gagnez uniquement si vous prenez les objets $3+3+3=9$.
- Sinon, Bob pourra lui aussi tout dépenser. Par ex. si vous prenez $2+2+2+3$.

Contrairement à l'exemple 3, il ne suffit pas prendre les plus petits objets.

Inversons les rôles !

Pendant la coding battle
Vous avez du résoudre les cas de tests à l'aveugle.

Après la coding battle
Vous avez du comprendre, adapter, ou améliorer les solutions proposées.

Et maintenant ?
Vous allez évaluer la qualité des cas de tests.

- Est-ce que la couverture de test est suffisante ?
- Est-ce que le score reflète bien la correction d'un algorithme ?

Et ensuite ?
Vous allez générer de meilleurs cas de test.

Évaluer les cas de test : faisons mentir l'oracle !

Algorithm: Meta Merch

```
1 S ← solve(I, W)
2 forall J ∈ S do
3     if solve(I − J, W) = ∅ then
4         return TRUE
5 return FALSE
```

Alternatives pour la méthode `solve`

H Méthode approchée pour le problème de décision.

D Méthode exacte pour le problème de décision.

E Méthode exacte pour le problème d'énumération.

Remarque sur le second appel de `solve` **(I.3)**
La résolution exacte du problème de décision est suffisante.

Zoom sur les alternatives à la méthode `solve`

Méthode approchée pour le problème de décision (H)
Programmez un algorithme glouton quelquonque : de nombreuses variantes sont possibles.

Méthode exacte pour le problème de décision (D)
Programmez une méthode de programmation dynamique avec une complexité pseudo-polynomiale : plusieurs variantes sont possibles.

Méthode exacte pour le problème d'énumération (E)

- Programmez un algorithme récursif, ou mieux un backtrack, de complexité exponentielle.
- Modifiez votre programme dynamique pour énumérer toutes les solutions avec une complexité pseudo-polynomiale.
  - Quelle est la complexité pour construire l'ensemble des solutions ?
  - Quelle est la complexité pour compter toutes les solutions ?

Protocole expérimental d'évaluation des cas de tests

Encodage des algorithmes candidats

On va construire plusieurs candidats en utilisant les alternatives à la méthode `solve` dans le méta-algorithme.

I.1	H	H	D	D	E	E
I.3	H	D	H	D	H	D

Protocole expérimental

- On résout les cas de test fournis par les organisateurs avec nos algorithmes candidats.
- On analyse les résultats présentés sous la forme de tableau en calculant les scores, et par des méthodes statistiques.

Protocole pour la génération de nouveaux cas de tests

1. On génère beaucoup de cas de tests aléatoires respectant les spécifications du problème avec les paramètres $Q \in \{10, 25, 50\}$, et $N \in \{10, 20, 30\}$,
2. On résout tous les cas de test avec tous les algorithmes candidats.
3. On analyse les résultats des algorithmes candidats.
4. On sélectionne entre 5 et 10 instances pour chaque valeur de $N \in \{10, 20, 30\}$.
5. On calcule le score de chaque algorithme candidat sur le jeu de test.

Questions ?