Enseignement en Programmation Sportive
EPS

TD2 - Premiers paradigmes classiques

Joël Felderhoff, Quoc Tung Le, Lucas Perotin, Eric Thierry

Debriefing : les verdicts du robot-juge

Jugements (les plus fréquents) :

- Accepted (AC) : Bonne réponse calculée dans les temps et dans l'espace imparti.
- Presentation Error (PE) : Réponse juste mais mal présentée : vérifier espaces, sauts de lignes, caractères cachés ...
- Wrong Answer (WA) : Réponse fausse.
- Compile Error (CE) : Erreur à la compilation.
- Runtime Error (RE) : Erreur à l'exécution (p.ex. segmentation fault ou division par zéro). Noter que le juge attend que le code termine en renvoyant 0.
- Time Limit Exceeded (TL) : Temps d'exécution dépassant la limite imposée par le juge. Ne donne aucune indication sur le fait que le programme aurait donné le bon résultat si on lui avait laissé le temps.

Debriefing : les verdicts du robot-juge

Jugements (les moins fréquents) :

- Memory Limit Exceeded (ML) : Espace d'exécution dépassant la limite imposée par le juge. Idem pas d'indication sur la correction du programme.
- Output Limit Exceeded (OL) : Taille de la sortie dépassant la limite imposée par le juge. Souvent un bug dans l'écriture en sortie de la réponse, comme une boucle infinie.
- Submission Error (SE) : Echec de l'étape de soumission du code au robot-juge. Souvent un problème de serveur ou de connection.
- Restriction Function (RF) : Détection dans le code soumis d'une fonction considérée comme nuisible au serveur ou comme de la triche, p.ex. faire exécuter une partie d'un algorithme pendant la phase de compilation.
- Can't Be Judged (CJ) :La base de donnée du juge est incomplète sur le problème choisi, il manque p.ex. l'entrée ou la réponse.

Briefing : structures de données (SD)

Structures de données utiles : objets mathématiques

- Mots : `string` en C++, `str` en Python
- Complexes : `complex` en C++, `complex` en Python

Structures de données utiles : conteneurs/containers

- Tableaux statiques : `array` en C++
- Tableaux dynamiques : `vector` en C++, `list` en Python
- Tableaux associatifs : `map` en C++, `dict` en Python
- Ensembles : `set` en C++, `set` en Python
- Tuples : `pair` et `tuple` en C++, `tuple` en Python

Utilisation :

- C++ : ne pas oublier de déclarer `#include<SD_utilisée>`
- Ne pas hésiter à utiliser les options du constructeur dès la déclaration

Briefing : premiers paradigmes classiques

Exemples de paradigmes :

- Brute force
- Glouton
- Programmation dynamique (qui porte mal son nom ... Smart Recursion serait mieux)
- Dichotomie (Divide Strategy)

Briefing : Brute force !

Enumération : quelques problèmes classiques

- Toutes les parties d'un ensemble de card n
- Toutes les parties de card k d'un ensemble de card n
- Toutes les permutations d'une liste de card n

Algorithmes d'énumération : $\exists$ routines déjà implémentées

- C++ : `next_permutation()` avec `#include<algorithm>`
- Python : `combinations()` et `permutations()` avec `import itertools`
- Bit Trick : Gosper's Hack

Backtracking : exploration des solutions potentielles par parcours d'un arbre de décision (choix consécutifs)

- Enumération parfois accélérée par troncage (élagage précoce)
- Ecriture souvent récursive

Briefing : Brute force !

Enumération : algo idéal génère N objets en temps $O(N)$, soit un temps amorti de $O(1)$ par objet (N pouvant être gros, p.ex. $N = 2^n$ ou $n!$ pour une taille d'entrée n).

Remarque : encore mieux si temps de génération $O(1)$ en pire cas pour chaque nouvel objet.

Exemples d'algos d'énumération :

- Gray codes
- Gosper's Hack

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

```
int set = (1<<k) - 1
// vecteur = k derniers bits à 1
while (set < (1<<n))
// on n'utilisera pas plus de n bits
// (1<<n = décalage de n bits à gauche, vaut 2^n)
{
int c = set & -set
// sélectionne dernier bit à 1 (seul bit à 1, le reste vaut 0)
// (& = ET bit à bit)
int r = set + c
// efface le dernier bloc de 1 et met un 1 devant
int s = r ^ set
// isole le dernier bloc de 1 de set (efface tout le reste)
// et rajoute donc un 1 devant (^ = XOR bit à bit)
int t = (s >> 2)/c
// décale ce bloc complètement à droite et décale encore de deux crans en plus
// cela revient à décaler complètement à droite le dernier bloc de 1 de set,
// en lui retirant aussi en 1 (on en a ajouté un et retiré deux)
set = t | r
// transforme donc set en avançant à gauche un 1 du dernier bloc de 1
// et en décalant tous les autres 1 complétement à droite
}
```

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$
\begin{aligned}
set &= 0^*0100111100 \\
-set &= \\
c = set \,\&\, -set &= \\
r = set + c &= \\
s = r \wedge set &= \\
t = (s >> 2)/c &= \\
set = t|r &= \\
set &= 0^*0101000111
\end{aligned}
$$

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$set = 0^*0100111100$$

$$-set = \overline{0^*0100111100} + 0^*0000000001$$

$$c = set \& -set =$$

$$r = set + c =$$

$$s = r \wedge set =$$

$$t = (s >> 2)/c =$$

$$set = t|r =$$

$$set = 0^*0101000111$$

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$set = 0^*0100111100$$

$$-set = 1^*1011000011 + 0^*0000000001$$

$$c = set \& -set =$$

$$r = set + c =$$

$$s = r \wedge set =$$

$$t = (s >> 2)/c =$$

$$set = t|r =$$

$$set = 0^*0101000111$$

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$set = 0^*0100111100$$
$$-set = 1^*1011000100$$
$$c = set \& -set =$$
$$r = set + c =$$
$$s = r \wedge set =$$
$$t = (s >> 2)/c =$$
$$set = t|r =$$
$$set = 0^*0101000111$$

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$
\begin{aligned}
set &= 0^*0100111100 \\
-set &= 1^*1011000100 \\
c = set \,\&\, -set &= 0^*0000000100 \\
r = set + c &= \\
s = r \wedge set &= \\
t = (s >> 2)/c &= \\
set = t|r &= \\
set &= 0^*0101000111
\end{aligned}
$$

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$set = 0^*0100111100$$
$$-set = 1^*1011000100$$
$$c = set \& -set = 0^*0000000100$$
$$r = set + c = 0^*0100111100 + 0^*0000000100$$
$$s = r \wedge set =$$
$$t = (s >> 2)/c =$$
$$set = t|r =$$
$$set = 0^*0101000111$$

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$
\begin{aligned}
set &= 0^*0100111100 \\
-set &= 1^*1011000100 \\
c = set \,\&\, -set &= 0^*0000000100 \\
r = set + c &= 0^*0101000000 \\
s = r \wedge set &= \\
t = (s >> 2)/c &= \\
set = t|r &= \\
set &= 0^*0101000111
\end{aligned}
$$

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$set = 0^*0100111100$$
$$-set = 1^*1011000100$$
$$c = set \& -set = 0^*0000000100$$
$$r = set + c = 0^*0101000000$$
$$s = r \wedge set = 0^*0001111100$$
$$t = (s >> 2)/c =$$
$$set = t|r =$$
$$set = 0^*0101000111$$

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$set = 0^*0100111100$$
$$-set = 1^*1011000100$$
$$c = set \& -set = 0^*0000000100$$
$$r = set + c = 0^*0101000000$$
$$s = r \wedge set = 0^*0001111100$$
$$t = (s >> 2)/c = 0^*0000000111$$
$$set = t|r =$$
$$set = 0^*0101000111$$

Briefing : l'algo du jour

Algo du jour : Gosper's Hack (k parmi n via indicatrice)

Exemple : 5 parmi 10 (0^* = complétion à 64 bits)

$$
\begin{aligned}
set &= 0^*0100111100 \\
-set &= 1^*1011000100 \\
c = set \,\&\, -set &= 0^*0000000100 \\
r = set + c &= 0^*0101000000 \\
s = r \wedge set &= 0^*0001111100 \\
t = (s >> 2)/c &= 0^*0000000111 \\
set = t|r &= 0^*0101000111 \\
set &= 0^*0101000111
\end{aligned}
$$