# Enoncé 1

Soit une liste T composée de N entiers positifs et soit un ensemble R d'opérations arithmétiques, avec R ⊆ {+, −, *, /, %} (% désigne le reste d'une division). Considérons les expressions arithmétiques, construites avec les éléments de T, les opérations de R et des parenthèses, qui satisfont aux conditions supplémentaires suivantes :

- chaque élément de T peut être utilisé une fois maximum,
- la division doit donner des résultats entiers (par exemple, 5/3 n'est pas autorisé),
- la soustraction doit donner des résultats positifs (par exemple, 3 − 5 n'est pas autorisé).

Par gap(T, R), on désigne le plus petit entier positif qui ne peut pas être le résultat de ces expressions arithmétiques.

Par exemple, supposons T = [1, 3, 4] et R = {+, −, *}. Quelle est la valeur de gap(T, R) ?

| | |
|---|---|
| 1 = 1 = 4 - 3 | 2 = 3 - 1 |
| 3 = 3 = 4 - 1 | 4 = 1 + 3 = 4 |
| 5 = 4 + 1 | 6 = 3 + 4 - 1 |
| 7 = 3 + 4 | 8 = 4 * (3 - 1) = 1 + 3 + 4 |
| 9 = (4 - 1) * 3 | |

Le plus petit entier positif qui ne peut pas être le résultat d'une expression construite à partir des éléments de T, R et des parenthèses est le nombre 10.

Soient deux entiers positifs M et N. Considérons toutes les listes possibles T contenant N entiers dans l'intervalle [1..M] et recherchons la valeur maximum de gap(T, R), notée gap(M, N, R).

Par exemple, gap(5, 2, {+, −, *}) = 5. Cette valeur est atteinte pour T = [1, 3].

Écrire une fonction:

int find_the_gap(int M, int N, char *R);

telle que, étant donnés des entiers M et N et une chaîne de caractères représentant R, retourne gap(M, N, R). La chaîne R se compose de caractères représentant des opérations arithmétiques: '+', '−', '*', '/' et/ou '%'; chaque caractère apparaît au maximum une fois.

Par exemple, find_the_gap(5, 2, "+ * −") doit retourner 5, comme expliqué plus haut.

Supposons que:

- M est un entier dans l'intervalle [1..10] ;
- N est un entier dans l'intervalle [1..4] ;
- La chaîne R se compose uniquement des caractères suivants: +, -, *, / et/ou %.

# Enoncé 2

Cette année, les Jeux Olympiques Intergalactiques auront lieu sur la planète appelée Terico. Votre entreprise, le leader mondial de l'affichage publicitaire, a eu l'idée d'installer une bannière à la surface de la lune de Terico.

Vous avez le droit de déployer une bannière rectangulaire. Les revenus tirés de la bannière seront proportionnels à sa superficie. Vous avez estimé le taux moyen de revenu généré par unité d'aire de la bannière.

Il faut tenir compte aussi de coûts d'infrastructure importants qui dépendent de la surface et de la nature du terrain sur lequel la bannière sera déployée. La surface de la lune a été divisée en cellules rectangulaires de même aire selon une grille cylindrique (nord-sud et est-ouest). Les M lignes de la grille, numérotées de 0 à M−1, correspondent aux parallèles (cercles de même latitude) de la lune, la première se situant près du pôle nord et la dernière près du pôle sud. Les N colonnes de la grille, numérotées de 0 à N−1, correspondent aux méridiens (lignes de même longitude) qui descendent du pôle nord au pôle sud. La grille est cylindrique, dans la mesure où la dernière colonne est adjacente à la première, et toute bannière rectangulaire qui recouvre cette transition est valide.

Pour chaque cellule de la grille, on connait le coût de couverture par la bannière qui y est associé. Vous pouvez déployer une bannière rectangulaire sur tout sous-rectangle de la grille. Le coût total de la bannière sera égal à la somme des coûts associés aux cellules couvertes, et les revenus seront proportionnels au nombre de cellules couvertes. Votre tâche consiste à trouver la taille et la position de la bannière qui maximisent les bénéfices (total des revenus moins coût total), et à déterminer la surface de la bannière (en nombre de cellules).

Une matrice C d'entiers, de taille M par N et indicée à partir de zéro, vous est donnée. C[I][J] correspond au coût de couverture associé à la cellule de la grille à la ligne I et à la colonne J. Un taux de revenu R, qui correspond à l'estimation des recettes par cellule générées par votre bannière, vous est également donné. Écrivez une fonction :

```
int banner(int **C, int M, int N, int R);
```

qui, à partir des paramètres C (matrice indicée à partir de zéro) et R (taux de revenu), renvoie la surface de la bannière qui maximise les bénéfices. S'il n'existe pas de bannière rentable (bénéfices ≤ 0 quelle que soit la bannière), la fonction doit renvoyer 0. S'il existe plusieurs bannières optimales, la fonction doit renvoyer la superficie de la plus petite d'entre elles.

Considérez par exemple la matrice C suivante, composée de trois lignes et quatre colonnes:

| | | | |
|---|---|---|---|
| C[0][0] = 2 | C[0][1] = 2 | C[0][2] = 2 | C[0][3] = 2 |
| C[1][0] = 1 | C[1][1] = 1 | C[1][2] = 3 | C[1][3] = 1 |
| C[2][0] = 2 | C[2][1] = 1 | C[2][2] = 3 | C[2][3] = 2 |

et le taux de revenu R = 2. Les cellules dont le coût est 1 sont rentables, celles de coût 3 sont déficitaires et celles de coût 2 sont neutres. Il existe deux rectangles optimaux. Le premier possède son coin supérieur gauche sur la ligne 0 et la colonne 3, et son coin inférieur droit sur la ligne 2 et la colonne 1. Le second a son coin supérieur gauche sur la ligne 1 et la colonne 3, et son coin inférieur droit sur la ligne 2 et la colonne 1. Ce dernier est celui qui possède l'aire la plus petite ; la fonction devrait donc renvoyer 6.

Considérez maintenant la matrice C suivante, composée de trois lignes et quatre colonnes:

| | | | |
|---|---|---|---|
| C[0][0] = 5 | C[0][1] = 2 | C[0][2] = 3 | C[0][3] = 2 |
| C[1][0] = 7 | C[1][1] = 4 | C[1][2] = 2 | C[1][3] = 3 |
| C[2][0] = 2 | C[2][1] = 8 | C[2][2] = 2 | C[2][3] = 3 |

et le taux de revenu R = 1. Aucune bannière rentable n'est réalisable, la fonction devrait donc renvoyer 0.

Contraintes:

- M et N sont des entiers compris entre 1 et 150;
- chaque élément de la matrice C est un entier compris entre 0 et 1 000 000;
- R est un entier compris entre 0 et 1 000 000.

Complexité:

- la complexité en temps dans le pire des cas est $O(M^3+N^3)$;
- la complexité en espace dans le pire des cas est O(M*N).