1 Entrée en scène à la Beckett

Dans sa pièce *Quad*, S. Beckett fait entrer et sortir, un par un, les n personnages de sorte que, la scène étant initialement vide, tous les sous-ensembles possibles (il y en a 2^n) de personnages apparaissent exactement une et une seule fois sur la scène. En notant $+i$ l'entrée en scène du personnage i et $-i$ sa sortie de scène, un scénario pour quatre personnages est par exemple $+1, +2, -1, +3, +1, -2, -1, +4, +1, +2, -1, -3, +1, -2, -1$. Dans la suite, on représentera un sous-ensemble de personnage comme un mot binaire dans lequel le bit d'indice i est à `1` si le personnage est présent sur scène et à `0` sinon. Ainsi, le problème est d'énumérer les entiers dont l'écriture en binaire est de longueur n de sorte que d'un entier à l'autre on ne modifie qu'un seul chiffre à la fois (on remarquera au passage qu'énumérer les entiers en ajoutant 1 à chaque fois ne produit pas une suite avec une telle propriété).

1. Donner la suite des entiers correspondant à l'exemple ci-dessus.
2. Montrer que si l'on connaît une solution S_n pour un n donné, une solution S_{n+1} pour $n+1$ personnages peut être obtenue de la manière suivante :
   (a) ajouter le caractère `0` à la fin (ou au début) de chaque mot de la solution S_n.
   (b) renverser la suite des mots engendrés par S_n et ajouter le caractère `1` à la fin (ou au début, même choix que précédemment) de chaque mot.
   (c) concaténer les deux listes obtenues.
3. Construire entièrement à la main deux solutions pour S_4 avec le procédé ci-dessus décrit.
4. Écrire la fonction `genFin` permettant de réaliser le procédé ci-dessus (version on ajoute à la fin).
5. Expliquer en quoi le procédé ci-dessus est correct.

2 Coefficients binomiaux

1. Rappeler la définition récursive du calcul des coefficients binomiaux défini pour les entiers naturels n et k.
2. Écrire une version récursive `binomial` du calcul du coefficient binomial de n et k.
3. Combien d'appels à la fonction `binomial` sont-ils effectués si l'on appelle `binomial(3,4)` ? Généraliser sur n et k.
4. Utiliser la technique de programmation dynamique étudiée en cours pour écrire `binomialRapide` améliorant le calcul du coefficient binomial de deux entiers.
5. Combien d'appels à `binomialRapide` sont-ils effectués si l'on appelle `binomialRapide(3,4)` ? Généraliser sur n et k.

3 Sous-suites maximales

Une *sous-suite* d'un mot u est un mot w obtenu à partir de u en effaçant des lettres et en respectant l'ordre. Par exemple, ara est une sous-suite des mots $quadratique$, $marchandage$ et $alcatraz$, mais n'est pas une sous-suite de $radar$.
On cherche une méthode qui retourne la longueur maximale d'une sous-suite commune à deux mots u et v donnés.

1. Dans un premier temps, on cherche à en donner une version récursive. On note $LSSM(i,j)$ la longueur maximale d'une sous-suite commune au mot formé des i premières lettres de u avec celui formé des j premières lettres de v.
   Que vaut $LSSM(i,0)$? $LSSM(0,j)$?
   On note u_i la i-ème lettre du mot u. Supposons que $u_i = v_j$.
   Exprimer $LSSM(i,j)$ en fonction de $LSSM(i-1,j-1)$.

Supposons maintenant que $u_i \neq v_j$.
Exprimer $LSSM(i,j)$ en fonction de $LSSM(i-1,j)$ et de $LSSM(i,j-1)$.
En déduire une méthode récursive naïve pour le problème. Pour $u = abac$ et $v = cbc$, dessiner l'arbre des appels récursifs et donner les états successifs de la pile.

2. Implémenter une version itérative puis une version récursive *dynamique* de cette méthode, en mémorisant les valeurs des sous-problèmes dans un tableau à deux dimensions. Construire le tableau pour les deux versions lorsque $u = abac$ et $v = cbc$.

4 Gain maximum

On dispose d'un jeton et d'une grille carrée dont chaque case porte un entier positif (on utilisera une variable globale `grille` de type tableau de tableaux d'entiers). On déplace le jeton depuis une case du bord inférieur vers une case du bord supérieur, en respectant la règle suivante. À chaque étape, on peut placer le jeton sur l'une des cases suivantes :

- celle juste au-dessus,
- celle située une position plus haut et une position plus à gauche (à condition que le jeton ne soit pas déjà dans la colonne la plus à gauche),
- celle située une position plus haut et une position plus à droite (à condition que le jeton ne soit pas déjà dans la colonne la plus à droite).

Chaque fois que l'on passe par une case, on reçoit la somme indiquée par l'entier associé.

1. Écrire une méthode auxiliaire `gainMax` qui prend en argument un indice de ligne `i` et un indice de colonne `j` et qui renvoie le gain maximum quand on part de la case `(i,j)` de `grille`.
2. En déduire une méthode `gainMax` sans argument qui renvoie le gain maximum pour `grille`.
3. En dessinant l'arbre des appels, donner la trace de cette méthode pour la grille suivante :

1	2	3
6	5	4
7	8	9

4. Calculer le nombre d'appels récursifs dans le cas général.
5. Préciser comment améliorer ce programme en utilisant le concept de programmation dynamique.

5 Calcul d'itinéraire

Vous êtes programmeur chez MaPie, société spécialisée dans le calcul d'itinéraire. La société permet de calculer le plus court chemin entre deux villes de France (on rappelle qu'il y a environ 36 000 communes en France, on supposera qu'elles sont numérotées de 0 à 35 999 dans la suite). Malin, vous récupérez sur Internet de quoi calculer le plus court chemin entre deux villes données à l'aide de la fonction `plusCourt(int v1,int v2)`. Le seul problème est que la fonction qui calcule l'itinéraire est relativement longue en temps de calcul. Comme vous envisagez de contribuer à la fortune des dirigeants et des actionnaires de la société pour laquelle vous travaillez, et que vous vous souvenez un peu des cours dispensés autrefois à l'Université, vous pensez utiliser un mécanisme proche de la programmation dynamique pour améliorer le temps de réponse moyen du service. Quoi ? Comment ? Écrivez la fonction `plusCourtRapide(int v1,int v2)`.
L'Europe contient environ 100 000 communes. Peut-on encore espérer agir ainsi ?