Questions : Chapitre 8

Question # 1

Lequel des algorithmes suivants est le plus efficace pour calculer le coefficient binomial $C(n,k) = \binom{n}{k}$?

A) Utiliser la formule

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

B) Utiliser la formule

$$C(n,k) = \frac{n \times (n-1) \times \ldots \times (n-k+1)}{k!}$$

C) Appliquer récursivement la formule

$$C(n,k) = C(n-1,k-1) + C(n-1,k) \text{ for } n \geq k > 0$$

$$C(n,0) = C(n,n) = 1$$

D) L'algorithme de programmation dynamique. Voir Algorithme 1.

E) Appliquer récursivement la formule

$$C(n,k) = \begin{cases} 1 & \text{si } k = 0 \\ \frac{n}{k} C(n-1,k-1) & \text{sinon} \end{cases}$$

Algorithme 1 : Binomial(n,k)

```
1 pour i = 0..n faire
2     pour j = 0.. min(i,k) faire
3         si j = 0 ou j = i alors
4             C[i,j] ← 1
5         sinon
6             C[i,j] ← C[i-1,j-1] + C[i-1,j]
7 retourner C[n,k]
```

Question # 2

Considérons deux équipes A et B qui jouent une série de parties jusqu'à ce qu'une équipe ait gagné n parties. En supposant que la probabilité que l'équipe A gagne une partie est la même pour toutes les parties et est égale à p et que la probabilité que l'équipe A perde une partie est $q = 1 - p$ (donc il n'y a jamais de partie nulle). Soit $P(i,j)$, la probabilité que A gagne la série si A a besoin de i victoires pour gagner alors que B a besoin de j victoires pour gagner.

A) Construisez une relation de récurrence pour $P(i,j)$ qui peut être utilisée par un algorithme de programmation dynamique.

B) Écrivez un algorithme pour résoudre ce problème selon l'approche de programmation dynamique et déterminez son efficacité en temps et en espace mémoire.

Question # 3

De façon générale, comment peut-on utiliser le tableau construit par l'algorithme de programmation dynamique pour savoir s'il y a plus d'une solution optimale pour le problème du « sac à dos » ?

Vous pouvez vous aider de l'instance du problème du « sac-à-dos » donné par le tableau 1 et de la trace de l'algorithme donnée par le tableau 2 lorsque la capacité du sac est de $W = 6$.

item	poids	valeur
1	3	25
2	2	20
3	1	15
4	4	40
5	5	50

TABLEAU 1 – Instance du problème du « sac à dos »

$i \backslash j$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	25	25	25	25
2	0	0	20	25	25	45	45
3	0	15	20	35	40	45	60
4	0	15	20	35	40	55	60
5	0	15	20	35	40	55	65

TABLEAU 2 – Solutions pour une instance du problème du « sac à dos »

Question # 4

Élaborez un algorithme de programmation dynamique pour le problème « rendre la monnaie » (étant donné un montant n et une quantité suffisante de pièces pour chacune des valeurs $D[1], D[2], \ldots, D[m]$, il faut trouver le plus petit nombre de pièces dont la somme est n ou indiquer que le problème n'a pas de solution). L'algorithme doit d'abord calculer le nombre de pièces de la solution optimale, puis retourner le nombre de pièces de chaque type utilisé dans cette solution.

Question # 5

Vous avez une image, que l'on peut encoder comme une matrice de pixels $A[1..m, 1..n]$, et vous désirez la compresser à l'aide du recadrage intelligent. Cette technique consiste à retirer un unique pixel sur chacune des m lignes de la matrice A. Pour éviter au maximum la corruption de l'image, les pixels supprimés sur deux lignes consécutives doivent être dans la même colonne ou dans des colonnes adjacentes de A. La suite de pixels retirés est appelée «couture».

Supposons que l'on vous fournit une matrice $P[1..m, 1..n]$ contenant les perturbations entraînées par le retrait de chaque pixel. Alors, vous avez que $P[i, j]$ est la perturbation encourrue si on retire le pixel $A[i, j]$. La perturbation totale est la somme des perturbations de la couture.

A) Construisez une relation de récurrence pour $T[i, j]$, la valeur minimale de la perturbation encourrue en retirant le pixel $A[i, j]$.

B) Élaborez un algorithme pour déterminer la valeur de la perturbation totale minimale pour une image donnée $A[1..m, 1..n]$.

C) Élaborez un algorithme qui détermine quels pixels doivent être retirés à partir de la solution retournée en A).

Question # 6

Une planche de longueur L doit être coupée aux positions $p = [p_1, p_2, \ldots, p_n]$. La planche doit passer dans une machine et ressort coupée à un seul endroit. Le temps pour couper une planche est proportionel à sa longueur.

A) Trouvez la relation de réccurence pour obtenir le temps minimum pour couper la planche.

B) Écrivez un algorithme pour résoudre ce problème selon l'approche de programmation dynamique.

C) Élaborez un algorithme pour énumérer l'ordre des coupes.