Université de Lille

FST- Département Informatique

ACT
Durée 1h
Tous documents autorisés

2017–2018
Master1 Informatique
23 octobre 2017

Nom : **Prénom :** **Groupe :**

Les algorithmes seront écrits en pseudo-langage ou dans un langage de votre choix. La clarté de vos réponses et de votre code sera prise en compte.

Soit une tablette de chocolat **carrée**, représentée par une grille de n colonnes et n lignes. Pour fixer les idées, le carré en haut à gauche est le carré de coordonnées $(0,0)$, tandis que le carré en bas à droite est le carré de coordonnées $(n-1,n-1)$. Certains carrés, les `carrés délicieux`, sont marqués d'un signe spécial (ici un cœur). On cherche une tablette carrée de taille maximum qui ne contienne que des carrés délicieux. Par exemple, pour la tablette ci-contre, la largeur maximum d'une tablette carrée qui ne contienne que des carrés délicieux est 4 et il y en a une dont le coin en haut à gauche est à la position $(1,2)$.

♡			♡	♡	♡	
		♥	♥	♥	♥	
♡		♥	♥	♥	♥	♡
	♡	♥	♥	♥	♥	
♡		♥	♥	♥	♥	♡
♡	♡	♡	♡	♡	♡	
	♡	♡		♡	♡	

Le problème est donc :

Entrée : une plaquette carré de côté n où certains carrés sont marqués "délicieux", représentée par un tableau T à n lignes et n colonnes où les coefficients valent 0 ou 1, 1 correspondant aux carrés délicieux.

Sortie : la largeur maximum k d'un carré délicieux dans la plaquette.

Pour l'exemple précédent, la sortie sera donc 4.

Q 1. Pour résoudre le problème, le professeur Leonidas propose la méthode suivante :

Q 1.1. Il propose d'abord la fonction :

```
\\ 0 <=l<n-k, 0 <= c<n-k, 0<=k
boolean test(int l; int c; int k){
   for (int i=l;i<l+k;i++)
     for (int j=c;j<c+k;j++)
        if (T[i][j]==0) return false;
   return true;}
```

Que calcule cette fonction ? Quel est l'ordre de grandeur de sa complexité en fonction de k ?

Retourne vrai si il existe un carré de taille k à la position (l,c), faux sinon. On peut se limiter à compter les tests du type (T[i][j]==0). Le nb de ceux-ci est k^2 dans le pire des cas.

Q 1.2. Il propose ensuite la fonction :

```
\\ 0 <=l<n-k, 0 <= c<n-k, 0<=k
boolean existDelice(int k){
   for (int i=0;i+k<=n;i++)
      for (int j=0;j+k<=n;j++)
        if (test(i,j,k)) return true;
   return false;}
```

Que calcule cette fonction ? Quel est l'ordre de grandeur de sa complexité en fonction de k et n ?

Retourne vrai SSi il existe un carré de taille k .
Le nb de tests est $k^2 * (n-k)^2$ dans le pire des cas

Q 1.3. Ensuite, pour calculer la largeur maximale d'une tablette "délicieuse", il propose :

```
int maxDelice(){
  for (int k=n;k>0;k--)
     if (existDelice(k)) return k;
  return 0;}
```

Quel est l'ordre de grandeur de la complexité de cette fonction en fonction de n ? Justifier brièvement.

Le nb de tests N est $\sum_{k=1}^{n} k^2 * (n-k)^2$ dans le pire des cas. On peut en déduire que $N \leq n^5$: c'est en $O(n^5)$. On peut remarquer que $N \geq \sum_{k=n/3}^{2n/3} k^2 * (n-k)^2$ soit $N > (n/3)^5$ donc c'est en $\Theta(n^5)$.

Très peu ont bien traité cette question, laissant k dans la réponse finale. Je n'ai mis le maximum des points que si il y avait une justification que c'était bien un $\Theta(n^5)$.

Q 1.4. Pour améliorer la complexité obtenue, le professeur propose de modifier la fonction précédente en utilisant la dichotomie pour la recherche de "k". Qu'en pensez-vous ? Justifier brièvement en donnant l'idée de l'algorithme et sa complexité.

Rep :

```
g=0;d=n;
while g <d {
 m=g+d/2;
 if existDelice(m)  g=m; else d=m-1;}
return g;
```

L'algo est en $\Theta(n^4 logn)$

Q 2. L'objectif est maintenant de proposer une méthode plus efficace en utilisant la programmation dynamique. Appelons $opt(l, c)$ la largeur maximale d'un carré délicieux à la position (l, c). L'idée est donc de remplir une table contenant les valeurs de $opt(l, c)$.

Q 2.1. Pouvez-vous finir de remplir la table pour l'exemple :

	0	1	2	3	4	5	6
0	1	0	0	3	2	1	0
1	0	0	4	3	2	1	0
2	1	0	4	3	2	1	1
3	0	1	3	3	2	1	0
4	1	0	2	2	2	1	1
5	1	2	1	1	2	1	0
6	0	1	1	0	1	1	0

Q 2.2. Si $T(l, c)$ n'est pas un carré délicieux, que vaut $opt(l, c)$?

Rep :

0

Q 2.3. Quelle est la largeur d'un carré maximal délicieux à une position de type $(n - 1, c)$ ou $(l, n - 1)$?

Rep :

1 si le carré est délicieux, 0 sinon

Q 2.4. Soit $0 \leq l < n - 1$ et $0 \leq c < n - 1$. Supposons qu'à la position (l, c) il y ait un carré délicieux de largeur k, avec $k \geq 2$. Exprimer $opt(l, c)$ en fonction de $opt(l + 1, c), opt(l, c + 1)$ et $opt(l + 1, c + 1)$.

Rep :

$opt(l, c) = min(opt(l + 1, c), opt(l, c + 1), opt(l + 1, c + 1)) + 1.$

Q 2.5. Déduire de ce qui précède un algorithme en $\Theta(n^2)$ pour résoudre le problème.

```
VOPT=0;
for (int x=0; x<n ; c++) {
OPT[n-1][x]=T[n-1][x];
OPT[x][n-1]=T[x][n-1];}
for (int l=n-2; l>0 ; l--)
 for (int c=n-2; c>0 ; c--)  {
   if (T[l][c]) OPT[l][c]=1+min(OPT[l+1][c], min(OPT[l+1][c+1], OPT[l][c+1]));
   else OPT[l][c]=0;
   VOPT=max(VOPT,OPT[l][c]);}
```

"erreurs" faites : oublis d'initialisation pour la dernière ligne ou colonne, remplissage dans le mauvais sens, implémentation sans mémoisation (donc complexité demandée non respectée).