CPE Lyon - 3ICS - Année 2020/21

Développement informatique - 1

TP 4 – Algorithmes gloutons, programmation dynamique

(travail individuel, temps estimé : 8h)

1/ Travail à faire

Vous allez devoir implémenter différents algorithmes répondant aux notions abordées dans le cours. Votre projet doit être versionné et votre code doit être structuré (utilisation de .h).

1.1 Exercice 1 : Divide and conquer.

Des volumes importants de données sont susceptibles d'être traitées par les ordinateurs. Des algorithmes efficaces sont alors nécessaires pour réaliser ces opérations comme, par exemple, la sélection et la récupération des données. Les algorithmes de recherche entrent dans cette catégorie. Leur rôle est de déterminer si une donnée est présente et, le cas échéant, d'en indiquer sa position, pour effectuer des traitements annexes. La recherche d'une information dans un annuaire illustre cette idée. On cherche si telle personne est présente dans l'annuaire afin d'en déterminer l'adresse. Plus généralement, c'est l'un des mécanismes principaux des bases de données : à l'aide d'un identifiant, on souhaite retrouver les informations correspondantes. Dans cette famille d'algorithmes, la recherche dichotomique permet de traiter efficacement des données représentées dans un tableau de façon ordonnée.

L'idée centrale de cette approche repose sur l'idée de réduire de moitié l'espace de recherche à chaque étape : on regarde la valeur du milieu et si ce n'est pas celle recherchée, on sait qu'il faut continuer de chercher dans la première moitié ou dans la seconde. Plus précisément, en tenant compte du caractère trié du tableau, il est possible d'améliorer l'efficacité d'une telle recherche de façon conséquente en procédant ainsi :

1. On détermine l'élément m au milieu du tableau ;
2. Si c'est la valeur recherchée, on s'arrête avec un succès ;
3. Sinon, deux cas sont possibles :
   (a) si m est plus grand que la valeur recherchée, comme la tableau est trié, cela signifie qu'il suffit de continuer à chercher dans la première moitié du tableau ;
   (b) sinon, il suffit de chercher dans la moitié droite.
4. On répète cela jusque avoir trouvé la valeur recherchée, ou bien avoir réduit l'intervalle de recherche à un intervalle vide, ce qui signifie que la valeur recherchée n'est pas présente.À chaque

étape, on coupe l'intervalle de recherche en deux, et on en choisit une moitié. On dit que l'on procède par dichotomie, du grec *dikha* (en deux) et *tomos* (couper).

Pour l'implémentation, vous respecterez la signature suivante :

```
/**
 * Recherche par dichotomie dans un tableau d'entiers
 * @param array The array of values
 * @param size_t  The size of the array
 * @param value  The value to find
 * @return The position of the value found  or -1
 */
int find_by_dichotomy(int array[], int size_t, int value )
```

1.2 Exercice 2 : Algorithme glouton.

Le problème du sac à dos, noté également KP (en anglais, Knapsack problem) est un problème d'optimisation combinatoire. Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou partie d'un ensemble donné d'objets ayant chacun un poids et une valeur. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum (https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_sac_%C3%A0_dos)

On se donne n objets ayant pour valeurs c_1 , … , c_n et pour poids (ou volume) w_1 , … , w_n. Le but est de remplir le sac à dos en maximisant $\sum_{i=1}^{n} c_i$, sous la contrainte $\sum_{i=1}^{n} w_i \leqslant W$ ou W est la contenance maximale du sac.

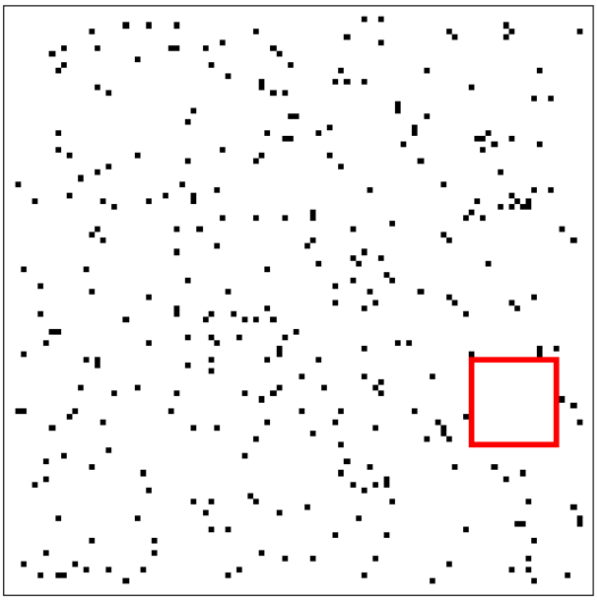
Pour l'approche « gloutonne » On travaillera sur le rapport "qualité/prix". On commence par trier les objets selon les $\frac{c_i}{w_i}$ décroissants, puis on gloutonne en remplissant le sac par le plus grand élément possible à chaque tour. Vous êtes invité à utiliser les structures pour stocker les couples c_i et w_i.

L'algorithme est-il optimal pour le jeu de valeurs suivant :

(W = 10) ($w_1 = 6$; $w_2 = 5$; $w_3 = 5$) ($c_1 = 7$; $c_2 = 5$; $c_3 = 5$)

1.3 Programmation dynamique : Problème du plus grand carré libre

On considère le problème suivant: étant donné une image monochrome n x n (ou une matrice), déterminer le plus grand carré blanc, i.e. qui ne contient aucun point noir.



Sous problème : Déterminer la taille PGCB(x,y) du plus grand carré blanc dont le pixel en bas à droite a pour coordonnées (x,y)

Observation clé : Un carré m x m de pixels C est blanc si et seulement si :

- le pixel en bas à droite de C est blanc ;
- Les trois carrés (m-1) x (m-1) en haut à gauche, en haut à droite et en bas à gauche sont tous blancs.

Relation de récurrence :

- Si le pixel (x,y) est noir, alors :

  PGCB(x,y)=0.

- Si (x,y) est blanc et dans la première ligne en haut ou la première colonne à gauche, alors :

  PGCB(x,y)=1.

- Si (x,y) est blanc et ni dans la première ligne en haut ni dans la première colonne à gauche, alors :

  PGCB(x,y) = 1 + min{PGCB(x-1,y-1),PGCB(x,y-1),PGCB(x-1,y)}.

Il est possible d'implémenter cet algorithme avec une approche récursive + mémoïsation ou une approche « du bas vers le haut ».

Pour cet exercice vous devez :

-Implémenter une fonction qui crée une matrice de taille n $_{x}$ n contenant des 0. Cette fonction remplira N cellules tirées au hasard avec la valeur 1 ;

-Implémenter une fonction qui affiche la matrice (1→ *, 0→ espace).

-Rechercher le plus grand carré ne contenant pas de valeur 1

-Implémenter une fonction qui affiche a nouveau la matrice et le plus grand carré trouvé (1→ *, 0→ espace, #→ pixel du carré trouvé).

2/ Consignes de Travail

- Le projet C doit se nommer ProgDyn
- Le travail est a réalisé seul.
- Le code doit être versionné.
- Le code doit respecter les bonnes pratiques de programmation.
- Le code doit être structuré. Vous créerez plusieurs fichiers (.c et .h). La répartition du code dans les fichiers doit être pertinente.
-Le code doit être correctement indenté, commenté (selon les besoins) et systématiquement documenté/annoté. Vous pouvez utiliser les annotations au format javadoc ou au format doxygen.
- Vous devez fournir un Makefile « fait main » permettant de compiler votre code.