I Autour de la dichotomie (E3A 2019)

Voici une tentative de code d'une fonction C pour tester par dichotomie si un entier `e` se trouve dans un tableau d'entiers `T` :

```
1  bool dicho(int e, int* T, int n) {
2          // Pré-condition : T est un tableau de n entiers trié dans l'ordre croissant
3          int g = 0, d = n - 1;
4          while (d - g > 0) {
5                  int m = (g + d) / 2;
6                  if (T[m] >= e) {
7                          d = m;
8                  } else {
9                          g = m;
10                 }
11             }
12         return T[g] == e;
13     }
```

1. Pour quelles raison ne remplace-t-on pas la précondition par un appel à une fonction qui trierait `T` dans l'ordre croissant?

> Solution : Cela augmenterait la complexité : passer de $O(\log_2(n))$ à $O(n\log_2(n))$ si le tri est en $O(n\log_2(n))$, par exemple. Notons qu'un tri peut être en $O(n\log_2(n))$ même sur un tableau déjà triée.

2. Montrer que la fonction `dicho` ne termine pas forcément.

> Solution : `dicho([0, 1], 1)` ne termine pas ($g=0, d=1, m=0$ et remplacer g par m ne change rien).

3. Indiquer sans justification la ou les corrections à apporter pour que la fonction `dicho` termine, tout en restant correcte.

> Solution : Remplacer `g = m` ligne (13) par `g = m + 1` (si `T[m] < e`, on peut exclure `m` des indices à regarder).

4. Justifier que la fonction corrigée termine et est correcte.

> Solution : Montrons alors que $d-g$ décroît strictement à chaque itération du `while`.
> Considérons une itération du `while` :
>
> - Si `T[m] >= e` : on remplace d par m et $m-g = \left\lfloor \frac{g+d}{2} \right\rfloor - g \leq \frac{g+d}{2} - g = \frac{g+d-2g}{2} = \frac{d-g}{2} < d-g$ car $d-g>0$ (condition du `while`).
> - Si `T[m] < e` : on remplace g par $m+1$ et $d-(m+1) = d-(\left\lfloor \frac{g+d}{2} \right\rfloor + 1) < d - \frac{g+d}{2}$ (car $\left\lfloor \frac{g+d}{2} \right\rfloor > \frac{g+d}{2} - 1$). D'où $d-(m+1) < \frac{d-g}{2} < d-g$.
>
> $d-g$ est un entier diminuant strictement à chaque itération du `while` donc devient négatif à un certain moment et la boucle s'arrête : dicho termine.
> Montrer l'invariant de boucle $\mathcal{P}$: «l'entier e apparaît dans le sous-tableau `T[g: d+1]` des éléments de `T` d'indices g à d».
> Si `T[m] >= e`: si $\mathcal{P}$ était vrai, comme `T` est triée, `e` apparaît avant l'indice `m`. Donc `e` apparaît dans `L[g:m+1]` et $\mathcal{P}$ reste vrai en remplaçant `d` par `m` (à l'itération suivante).
> Si $\mathcal{P}$ était faux alors `e` n'apparaît pas dans `L[g:d+1]` donc n'apparaît pas non plus dans `L[g:m+1]`.
> Raisonnement similaire si `T[m] < e`. Donc dicho est correct.

Une première extension consiste à réduire le problème non pas en 2 mais en 3 : c'est le principe de trichotomie.

5. Écrire une fonction `bool tricho(int e, int* T, int n)` qui renvoie `true` si l'élément `e` se trouve dans `T` et `false` sinon.

> Solution : On découpe l'intervalle $[\![i,j]\!]$ en $[\![i,m_1]\!] \cup [\![m_1,m_2]\!] \cup [\![m_2,j]\!]$ où $m_1 = \left\lfloor i + \frac{j-i}{3} \right\rfloor = \left\lfloor \frac{2i+j}{3} \right\rfloor$ et $m_2 = \left\lfloor i + 2\times\frac{j-i}{3} \right\rfloor = \left\lfloor \frac{i+2j}{3} \right\rfloor$:

```
bool tricho(int e, int* T, int n) {
        int i = 0, j = n - 1;
        while (i < j) {
                int m1 = (2*i + j) / 3;
                int m2 = (i + 2*j) / 3;
                if (e <= T[m1]) {
                        j = m1;
                    } else if (e <= T[m2]) {
                        i = m1 + 1;
                        j = m2;
                    } else {
                        i = m2 + 1;
                    }
            }
        return T[i] == e;
    }
```

6. Estimer la complexité de `tricho(e, T, n)` en fonction de `n`. Comparer avec la méthode par dichotomie.

Solution : Soit $C(n)$ la complexité de `tricho(int e, int* T, int n)`. Soit K le nombre maximum d'opérations réalisées lors d'un appel à `tricho(int e, int* T, int n)` (sans compter les appels récursifs). Comme un appel récursif s'effectue sur une T de taille divisée par (au moins) 3 : $C(n) \overset{(*)}{\leq} K+C(\frac{n}{3}) \leq K+K+C(\frac{n}{9}) \leq ... \leq K\times p+C(\frac{n}{3^p})$, en appliquant p fois $(*)$.
Avec $p = \lceil \log_3(n) \rceil$, on trouve $C(n) = O(\log_3(n)) = \boxed{O(\log_2(n))}$ (tous les logarithmes sont égaux à une constante près car $\log_b(a) = \frac{\ln(a)}{\ln(b)}$).
La complexité en terme de O(...) est donc inchangée.
Remarque : si l'on s'intéresse au nombre exact d'opérations ce n'est pas clair : $\log_3(n) < \log_2(n)$ mais la constante cachée dans le O(...) augmente aussi (on fait plus de d'opérations élémentaires à chaque itération).

Une seconde extension consiste à adapter le principe de dichotomie au cas d'une matrice d'entiers dont les éléments sont triés en colonne de haut en bas et de gauche à droite. L'illustration ci-dessous indique l'ordre des éléments d'une matrice à 4 lignes et 6 colonnes :

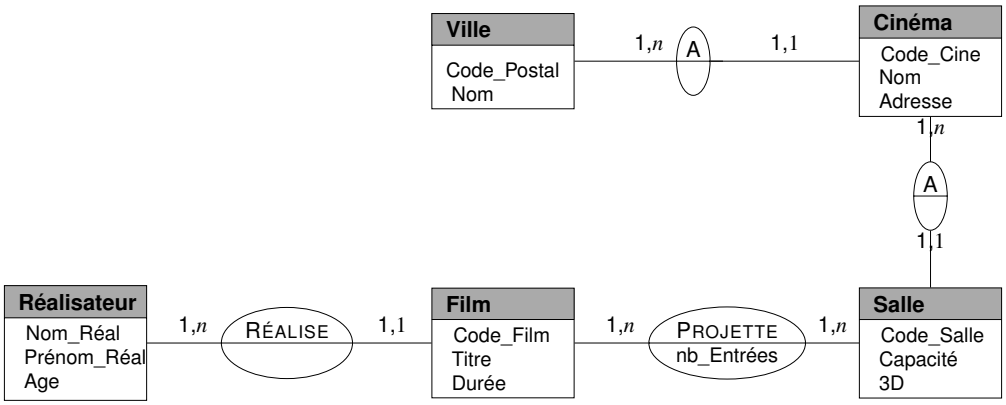
$$\begin{pmatrix} 0 & 4 & 8 & 12 & 16 & 20 \\ 1 & 5 & 9 & 13 & 17 & 21 \\ 2 & 6 & 10 & 14 & 18 & 22 \\ 3 & 7 & 11 & 15 & 19 & 23 \end{pmatrix}$$

7. Expliquer comment déterminer si un entier `e` se trouve dans une telle matrice de taille $n \times p$. , en complexité logarithmique en $\max(n, p)$.

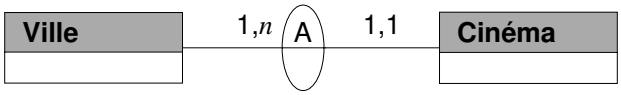
Solution : On effectue une première recherche par dichotomie pour trouver la bonne colonne, puis une autre recherche par dichotomie sur cette colonne pour trouver la bonne ligne. La complexité est O(log(p)) ($1^{\text{ère}}$ recherche dichotomique) + O(log(n)) ($2^{\text{ème}}$ recherche dichotomique) = $\boxed{\text{O}(\log(\max(n,p)))}$.

Partie II - Bases de données

On considère la base de données décrite par le modèle entité-association suivant :



Dans cette représentation, on lie des entités par des associations avec des cardinalités. Ainsi par exemple



peut se lire comme “Une ville a 1 à n cinéma(s)“ et “un cinéma est présent dans 1 et une seule ville“. Parfois, l'association possède des propriétés (le nombre d'entrées d'un film projeté dans une salle par exemple).

On suppose que le champ 3D de l'entité Salle est un entier, valant 0 si la salle n'est pas en 3D, et 1 sinon. On suppose de plus que la durée des films est en heures. Enfin, on affirme qu'une ville peut être uniquement déterminée par son code postal et son nom.

Q6. Donner le schéma relationnel correspondant. Préciser les clés primaires et étrangères des relations. Les clés primaires peuvent être associées à plusieurs attributs.

Q7. Écrire les requêtes suivantes en langage SQL :

(i). Donner le titre des films durant moins de 2h.

(ii). Donner le nom et le prénom du réalisateur ayant réalisé le film “Matrix”.

(iii). Compter le nombre de cinémas à Nantes.

(iv). Donner l'adresse des cinémas contenant au moins une salle 3D.

(v). Donner le code des films projetés dans toutes les salles.

(vi). Donner la liste des titres des films projetés dans le cinéma “Le Rio”.

Solution :

Q6. Il est demandé de définir les relations sous forme de table. On peut proposer :

- Ville(Code_Postal(int), nom(string, étrangère))
- Ville_A_Cine(Code_Postal(int, étrangère), Code_Cine(int, étrangère))
- Cine_A_Salle(Code_Cine(int, étrangère), Code_Salle(int, étrangère))
- Salle(Code_Salle(int), Capacité(int), 3D(int))
- Projette(Code_Salle(int, étrangère), Code_Film(int, étrangère))
- Film(Code_Film(int), Titre(string), Durée(int))
- Réalise(Code_Film(int, étrangère), Nom_Réal(string, étrangère), Prénom_Réal(string, étrangère))
- Réalisateur(Nom_Réal(string), Prénom_Réal(string), Age(int))

Q7. (a)

```sql
--- Donner le titre des films durant moins de 2h.
SELECT Titre FROM Film WHERE Durée < 2
```

(b)

```sql
--- Donner le nom et le prénom du réalisateur ayant réalisé le film "Matrix".
SELECT Nom_Réal, Prénom_Réal
FROM Réalise JOIN Film ON Réalise.Code_Film = Film.Code_Film
WHERE Titre = "Matrix"
```

(c)

```sql
--- Compter le nombre de cinémas à Nantes
SELECT COUNT(*)
FROM Ville_A_Cine JOIN Ville ON Ville_A_Cine.Code_Postal = Ville.Code_Postal
WHERE nom = "Nantes"
```

(d)

```sql
--- Donner l'adresse des cinémas contenant au moins une salle 3D
SELECT Adresse
FROM Cinéma JOIN Cinema_A_Salle ON Cinéma.Code_Cine = Cinema_A_Salle.Code_Cine
JOIN Salle ON Cinema_A_Salle.Code_Salle = Salle.Code_Salle
WHERE 3D = 1
```

(e)

```sql
--- Donner l'adresse des cinémas contenant au moins une salle 3D
SELECT Adresse
FROM Cinéma JOIN Cinema_A_Salle ON Cinéma.Code_Cine = Cinema_A_Salle.Code_Cine
JOIN Salle ON Cinema_A_Salle.Code_Salle = Salle.Code_Salle
WHERE 3D = 1
```

(f)

```sql
--- Donner le code des films projetés dans toutes les salles.
SELECT Code_Film
FROM Projette
GROUP BY Code_Film
HAVING COUNT(DISTINCT Code_Salle) = (SELECT COUNT(*) FROM Salle)
```

(g)

```sql
--- Donner la liste des titres des films projetés dans le cinéma "Le Rio".
SELECT Titre
FROM Film JOIN Projette ON Film.Code_Film = Projette.Code_Film
JOIN Salle ON Projette.Code_Salle = Salle.Code_Salle JOIN Cinema_A_Salle ON Salle.Code_Salle = Cinema_A_Salle.Co
JOIN Cinéma ON Cinema_A_Salle.Code_Cine = Cinéma.Code_Cine
WHERE Nom = "Le Rio"
```