TD 4 : Calculs de complexité

Amar AHMANE
MP2I

Exercice 3 : Couleur Majoritaire

Soit $n \in \mathbb{N}^*$. On pose $E = [\![1, n]\!]$. Soit $\mathbb{L}$ un ensemble dont les éléments sont des couleurs, on note alors R l'application définie par

$$R : \begin{array}{ccc} E & \to & \mathbb{L} \\ x & \mapsto & R(x) \end{array}$$

On fait alors quelques définitions :

(i) Soient $x \in E$, $c \in \mathbb{L}$. On dit que x est de couleur c si et seulement si $R(x) = c$.

(ii) On dit que deux entiers i et j de E sont "de même couleur" si et seulement si $R(i) = R(j)$.

(iii) Soit $c \in \mathbb{L}$, on note E_c l'ensemble des entiers dans E de couleur c. Alors

$$E_c = \{x \in E \mid R(x) = c\}$$

(iv) Soit $c \in \mathbb{L}$, on dit que c est majoritaire dans E si et seulement si $|E_c| > \dfrac{|E|}{2}$.

On peut considérer que cette portion de code est écrite en début de fichier, et que cette structure peut être utilisée dans les fonctions que l'on programmera dans la suite :

```c
struct ec {
        int couleur;
}
```

9 n désigne la taille du pointeur sur `ec`.

```c
int count(struc ec x, struct ec* ecs, int n){
        int result = 0;
        for(int i = 0; i < n; i++){
                if(ecs[i].couleur == x.couleur) result++;
        }
        return result;
}
```

Il s'agit d'un algorithme en $\Theta(n)$.

10 O

```c
bool has_max(struct ec* ecs, int n, int l){
        int* colours = (int*) malloc(l*sizeof(int));
        for(int i = 0; i < l; i++){
                colours[i] = -1;
        }
        for(int i = 0; i < n; i++){
                if(colours[ecs[i].couleur] == -1){
                        int occ_in_i = count(ecs[i], ecs, n);
                        if(occ_in_i > n/2) return true;
                }
        }
        return false;
}
```

11 Soient $c \in \mathbb{L}$, $k \in \mathbb{N}$ et $(E_i)_{i \in [\![1,k]\!]}$ une partition de E. Supposons que c est majoritaire dans E, montrons, par l'absurde, qu'il existe $i \in [\![1,k]\!]$ tel que c est majoritaire dans E_i.

Supposons alors que pour tout $i \in [\![1,k]\!]$, c n'est pas majoritaire dans E_i, i.e $E_{ic} \leq \frac{|E_i|}{2}$. En ce cas

$$\sum_{i=1}^{k} |E_{ic}| \leq \sum_{i=1}^{k} \frac{|E_i|}{2} \tag{1}$$

Or

$$\begin{aligned}\sum_{i=1}^{k} \frac{|E_i|}{2} &= \frac{1}{2} \sum_{i=1}^{k} \sum_{x \in E_i} 1 \\ &= {}^{1}\frac{1}{2} \sum_{x \in E} 1 \\ &= \frac{|E|}{2}\end{aligned}$$

Montrons que $(E_{ic})_{i \in [\![1,k]\!]}$ est un recouvrement disjoint de E_c.

(i) Soit $(i,j) \in [\![1,k]\!]^2$. On a $E_{ic} \subset E_i$ et $E_{jc} \subset E_j$, or $E_i \cap E_j = \emptyset$, donc $E_{ic} \cap E_{jc} = \emptyset$.

(ii)
$$\bigcup_{i=1}^{k} E_{ic} = \bigcup_{i=1}^{k} E_i \cap E_c = E_c \cap \bigcup_{i=1}^{k} E_i = E_c \cap E = E_c$$

Donc, on a

$$\begin{aligned}\sum_{i=1}^{k} |E_{ic}| &= \sum_{i=1}^{k} \sum_{x \in E_{ic}} 1 \\ &= {}^{2} \sum_{x \in E_c} 1 \\ &= |E_c|\end{aligned}$$

Donc d'après (1)

$$|E_c| \leq \frac{|E|}{2}$$

Autrement dit c n'est pas majoritaire dans E, ce qui est absurde.

12 L'idée est la suivante pour cet algorithme : les couleurs majoritaires dans chacun des E_i sont des couleurs candidates, du fait de l'implication prouvée plus haut. Le travail de l'algorithme sera ainsi d'établir une liste d'au plus k candidats, puis ensuite de vérifier s'il y en a un qui est majortaire dans E. Le tout se fait en $\mathcal{O}(n\sqrt{n})$.

```c
bool has_max_bis(struct ec* ecs, int k, int* bornes, int l){
        int* pmax = (int*) malloc(k*sizeof(int));
        int* colours = (int*) malloc(l*sizeof(int));
        int occ_in_i;
        int p = 0;
        for(int i = 0; i < k; i++){
                for(int j = 0; j < l; j++){
                        colours[i] = -1;
                }
                for(int j = bornes[i]; j <= bornes[i + 1]; j++){
                        if(colours[ecs[j].couleur] == -1){
                                occ_in_i = 0;
                                for(int k = bornes[i]; k <= bornes[i + 1; k++){
```

```
                        if(ecs[k].couleur == ecs[j].couleur) occ_in_i++;
                    }
                    if(occ_in_i > k/2) {
                        pmax[i] = ecs[j];
                        p++;
                        break;
                    }
                }
            }
        }
        for(int i = 0; i < p; i++){
            if(count(pmax[i], ecs, bornes[k] + 1) > (bornes[k] + 1)/2) return true;
        }
        return false;
    }
```

13 On continue d'utiliser l'implication prouvée en question 11 dans cette partie. La seule différence est que l'on possède ici, à chaque fois, au plus deux candidats. À chaque instance, les lignes 4 et 5 nous les fournissent, il suffit alors de faire la vérification. Celle-ci se faisant en temps n, on a que $T(n) \leq T(1) + n\log_2(n)$, donc $T(n) = \mathcal{O}(n \log n)$

```
1   struct ec max_dicho(struc ec* ecs, int first, int last){
2       struct ec default_x;
3       int mid = (first + last)/2;
4       int occ = 0;
5       struct ec x1 = max_dicho(ecs, first, mid);
6       struct ec x2 = max_dicho(ecs, mid + 1, last);
7       default_x.couleur = -1;
8       if(last == first){
9           return ecs[first];
10      } else if(last > first || first > last){
11          return default_x;
12      }
13      if(x1.couleur != -1){
14          for(int i = first; i <= last; i++){
15              if(ecs[i].couleur == x1.couleur) occ++;
16          }
17          if(occ > (first - last)/2) return x1;
18          occ = 0;
19      }
20      if(x2.couleur != -1){
21          for(int i = first; i <= last; i++){
22              if(ecs[i].couleur == x2.couleur) occ++;
23          }
24          if(occ > (first - last)/2) return x2;
25          free(occ);
26      }
27      return default_x;
28  }
```

14 Supposons que $n \geq 3$. Soient $x, y \in E$ deux entiers distincts de couleurs distinctes. Soit $c \in \mathbb{L}$, supposons que c est majoritaire dans E. Les ensembles $E \backslash \{x, y\}$ et $\{x, y\}$ forment une partition de E. Or, il n'existe pas de couleur majoritaire dans $\{x, y\}$, donc c est majoritaire dans $E \backslash \{x, y\}$.

1. $(E_i)_{i \in [\![1,k]\!]}$ est une partition de E.

1. $(E_{ic})_{i \in [\![1,k]\!]}$ est un recouvrement disjoint de E_c.