Algorithmique — M1 — 9/1/9 — corrigé

Examen du 9 janvier 2009

Université Paris Diderot

On applique les algorithmes de cours

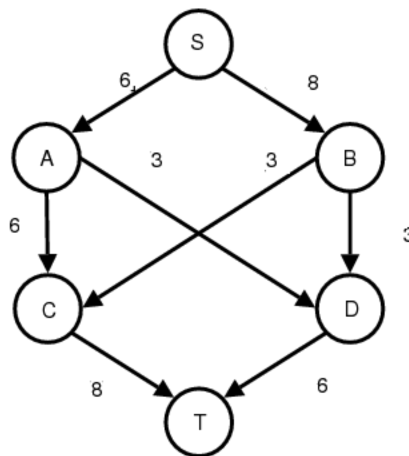
Exercice 1 – *Les reines*

Placer les 4 reines sur un tableau 4×4 en utilisant l'algorithme *backtracking* du cours. Sans énoncer l'algorithme montrez toutes les configuration considérées lors de son fonctionnement.

Correction. Une configuration consiste à placer les reines sur les colonnes de 1 à k (k=0..4). On la représente par un vecteur de k entiers qui correspondent aux lignes de ces reines

1
13
14
142
2
24
241
2413- trouvé

Exercice 2 – *Flux maximum*



Pour le réseau ci-dessus on cherche à trouver le flux (flot) maximum en appliquant un algorithme de cours.

1. Choisissez un algorithme (écrivez juste son nom s'il s'agit d'un algorithme connu).

   Correction. Ford-Fulkerson.

2. Appliquez l'algorithme (dessinez toutes ses itérations).
3. Donnez le résultat final : flux maximum et sa valeur.

Correction. voir la feuille scannée

On adapte un algorithme de cours

Exercice 3 – *Magasin*

Le nouveau magasin “Galeries Fulkerson" a n rayons $(r_1, r_2, \ldots, r_n)$. Pour travailler dans le magasin il y a $m \geq 2n$ vendeurs candidats $(v_1, v_2, \ldots, v_m)$. Les compétences des vendeurs sont représentées par une relation

$$C = \{(i, j) \mid \text{ vendeur } v_i \text{ peut travailler dans le rayon } r_j\}.$$

On cherche un algorithme qui décide s’il est possible d’embaucher $2n$ vendeurs et de les affecter aux rayons en respectant les conditions suivantes :
- chaque vendeur embauché est affecté à un rayon et un seul ;
- dans chaque rayon il y a exactement deux vendeurs ;
- chaque vendeur est compétent dans son rayon.

L’algorithme doit aussi proposer quels candidats embaucher et comment les affecter aux rayons.

1. Proposez un algorithme efficace pour ce problème.

Correction. On définit un réseau qui contient les sommets suivants :
–source S et T,
–un sommet pour chaque vendeur : $(v_1, v_2, \ldots, v_m)$,
–un sommet pour chaque rayon $(r_1, r_2, \ldots, r_n)$,
–target T,
et les arcs suivants :
–de S vers chaque v_i (capacité 1),
–de v_i vers r_j lorsque $(i, j) \in C$ (capacité 1),
–de chaque r_j vers T (capacité 2).
Ensuite on applique Ford-Fulkerson pour trouver un flux entier maximal. Si sa valeur $< 2n$ on répond "impossible". Si sa valeur est $2n$, alors on embauche v_i si le flux de S à v_i est non-nul. Lorsque le flux de v_i vers r_j est non-nul on affecte le vendeur i au rayon j.

2. Justifiez la correction de votre algorithme (donnez une ébauche de preuve).

Correction.
–**Si l’algo trouve une solution, elle est correcte** Effectivement, on embauche $2n$ vendeurs, et les contraintes de capacités et de préservation de flux nous garantissent tout ce qu’il faut :
 –Chaque vendeur embauché travaille dans un rayon et un seul (flux entrant de v égale 1, donc le flot sortant aussi).
 –Chaque vendeur travaille dans un rayon où il est compétent (sinon on aurait un flux interdit)
 –Le flux sortant de chaque r vaut 2 (nécessaire pour avoir la valeur 2n), donc le flux entrant aussi. Ça signifie qu’il y a 2 vendeurs affectés à chaque rayon.
–**Si une solution existe, l’algo trouve une solution**. Effectivement, à partir d’une solution pour le magasin il est possible de construire un flux de valeur 2n. Or, Ford-Fulkerson trouvera un flux optimal, qui aura forcément la valeur 2n aussi (par construction du réseau impossible de faire mieux que 2n). On conclut que notre algo trouve une solution.

3. Analysez sa complexité.

Correction. On a $V = n + m + 2$ sommets et $E = O(nm)$ arcs, on peut construire le réseau en $O(nm)$ opérations. La valeur n’excède pas $|f| = 2n$. Or, la complexité de Ford-Fulkerson est $O(|E|\ |f|)$, c’est - à dire $O(n^2m)$ ce qui est raisonnable.

On invente des algorithmes

Exercice 4 – *Le champion*

Les éléments d'un tableau donné $B = (b_1, b_2, \ldots, b_n)$ sont des objets dont on peut tester l'égalité, mais qu'on ne peut pas comparer (ou non plus classer). Le **champion** de B est l'élément présent dans le tableau strictement plus que n/2 fois.

1. Démontrer qu'un tableau peut contenir soit 0 soit 1 champion.

Correction. Supposons qu'un tableau contient 2 champions différents a et b (et éventuellement d'autres). Par définition de champion, y a plus de $n/2$ occurrences de a et plus $n/2$ occurrences de b - ça fait en tout plus de n objets dans un tableau de n. Contradiction. Donc on a toujours moins de 2 champions.

2. Programmez une fonction booléenne isChamp(x, B, s, f) qui teste est-ce que x est champion de $(b_s, \ldots, b_f)$. Indication : c'est très facile.

Correction. Voici un algo de complexité $O(n)$ (ou, plus précisément, $O(f - s)$)

```
Boolean isChamp(x, B,s,f)
        count=0
        pour i de s à f
            si x[i]=B
                    count++
        si 2*count>f+1-s
            retourner vrai
            sinon retourner faux
```

3. Proposez un algorithme itératif "naïf" qui trouve le champion ou répond qu'il n'y en a pas. Quelle est sa complexité ?

Correction. Il suffit de tester chaque objet avec la fonction précédente.

```
ChampNaif(B,s,f)
        pour i de s à f
            si isChamp(B[i],B,s,f)
                            retourner B[i]
        retourner null
```

Il y au pire n itérations de complexité $O(n)$ chacune, donc la complexité totale est $O(n^2)$

4. Proposez un algorithme plus efficace de type Diviser-Pour-Régner qui trouve le champion ou répond qu'il n'y en a pas.

Indication.

- On cherche à programmer la fonction $Champ(B, s, f)$ qui renvoie la valeur du champion de $(b_s, \ldots, b_f)$ ou null s'il n'y a pas de champion
- Coupez le tableau en deux moitiés.
- Chaque moitié peut avoir ou non son champion (il y a en tout 4 cas). Qu'est-ce qu'on peut affirmer sur le champion du grand tableau pour chacun de ces 4 cas. Justifiez vos réponses.
- Donnez un algorithme récursif pour $Champ(B, s, f)$.

Correction. L'observation clé est que le champion de tableau doit être aussi champion d'une de ses moitiés (peut-être des deux). Autrement dit les seuls candidats à tester dans le grand tableau sont les champions de la moitié gauche et le champion de la moitié droite. Ceci justifie l'algo DPR suivant :

```
Champ(B,s,f)
    si s=f
        retourner B[s]

    m=(s+f)/div 2
    chGauche=Champ(B,s,m)
```

```
si chGauche !=null ET IsChamp(chGauche,B,s,f)
    retourner chGauche
chDroite=Champ(B,m+1,f)
si chDroite !=null ET IsChamp(chDroite,B,s,f)
    retourner chDroite
retourner null
```

5. Analysez la complexité de votre algorithme Diviser-Pour-Régner.

Correction. La fonction fait deux appels récursifs et encore $O(1)$ opérations. Donc $T(n) = 2T(n/2) + O(n)$, ce qui correspond au cas moyen du Master Theorem, et la complexité est $O(n \log n)$.

Exercice 5 – *La monnaie*

On a un stock illimité de pièces de monnaie de chacune de m valeurs différentes $\alpha = p_1, p_2, \ldots, p_m$. On peut représenter certains montants A avec ces monnaies. Par exemple, pour les pièces de $2, 3, 5$ (centimes) et le montant $A = 11$ il existe des représentations suivantes (et d'autres - à la fin de l'exercice on saura combien)

$$\begin{aligned} 11 &= 2+2+2+5 \\ 11 &= 2+3+3+3 \end{aligned}$$

Le problème algorithmique à résoudre dans cet exercice est le suivant : étant donné $\alpha = p_1, p_2, \ldots, p_m$ et A trouver **le nombre de représentations** différentes (sans tenir compte de l'ordre) du montant A par les pièces α. On utilisera la programmation dynamique pour concevoir un algorithme qui résolve ce problème.

1. Soit $R(i, j)$ le nombre de représentations du montant j avec les i premières pièces $p_1, \ldots, p_i$ Écrivez les équations de récurrence pour cette fonction sans oublier les cas de base.

Correction. L'idée de la récurrence est la suivante : si $j \geq p_i$, alors soit on utilise la dernière pièce (et ensuite on fait la monnaie pour $j - p_i$), soit on ne l 'utilise pas. Dans le cas où $j < p_i$ la pièce p_i est inutile.

$$R(i,j) = \begin{cases} 1 & \text{si} \quad j = 0 \\ 0 & \text{sinon si} \quad i = 0 \\ R(i-1,j) & \text{sinon si} \quad j < p_i \\ R(i, j-p_i) + R(i-1,j) & \text{sinon} \end{cases}$$

Une petite optimisation possible mais optionnelle concerne le cas d'une seule pièce :

$$R(i,j) = \begin{cases} 1 & \text{si} \quad i = 1 \wedge j(\mod i = 0) \\ 0 & \text{si} \quad i = 1 \wedge (j \mod i \neq 0) \\ \ldots \end{cases}$$

2. Écrivez un algorithme efficace (récursif avec “marquage” ou itératif) pour calculer R .

Correction. On crée un tableau memo[0..m,0..A] pour mémoriser les résultats de nos calculs , et on le remplit de -1.
Ensuite on utilise la fonction suivante

```
int R(i,j)
si on l'a déjà calculé on retourne
    if memo[i,j]>-1 return memo[i,j]


sinon on calcule en utilisant les équations de récurrence
    if (j==0)
            R=1
    else if (i==0)
            R=0
    else if (i=1) AND j mod i =0
            R=1
    else if (i=1)AND j mod i !=0
            R=0
    else if (j<p[i])
            R= R(i-1,j)
    else
            R= R(i,j-p[i])+R(i-1,j)

on mémorise et on retourne
memo[i,j]=R
return R
```

3. En sachant calculer la fonction choisie R , comment répondre à la question initiale : trouver le nombre de représentations différentes du montant A par les pièces α.

Correction. Il suffit d'appeler R(m,A).

4. Analysez la complexité de votre algorithme.

Correction. L'initialisation de memo coûte $O(mA)$. La première visite de chaque case coûte O(1)+ 2 visites d'autres cases. Une re-visite coûte O(1). Toutes les premières visites ensemble coûtent $O(mA)$ plus encore $O(mA)$ de re-visites, ce qui donne O(mA).
On conclut que la complexité est O(mA).

5. Appliquez votre algorithme à l'exemple ci-dessus ($\alpha = 2, 3, 5$ et $A = 11$).

Correction.

$$
\begin{aligned}
R(3,11) &= R(3,6) + R(2,11) \\
R(3,6) &= R(3,1) + R(2,6) \\
R(3,1) &= R(2,1) = R(1,1) = R(0,1) = 0 \\
R(2,6) &= R(2,3) + R(1,6) \\
R(2,3) &= R(2,0) + R(1,3) = 1 + R(1,3) \\
R(1,3) &= 0 \\
R(1,6) &= 1 \\
R(2,11) &= R(2,8) + R(1,11) = R(2,8) + 0 \\
R(2,8) &= R(2,5) + R(1,8) = R(2,5) + 1 \\
R(2,5) &= R(2,2) + R(1,5) = R(2,2) + 0 \\
R(2,2) &= R(1,2) = 1
\end{aligned}
$$

On remonte

$$\begin{aligned}
R(2,5) &= 1 \\
R(2,8) &= 2 \\
R(2,11) &= 2 \\
R(2,3) &= 1 \\
R(2,6) &= 2 \\
R(3,6) &= 2 \\
R(3,11) &= 4
\end{aligned}$$

On a trouvé la réponse : 4.