Algorithmique — M

Examen du [illegible] -corrig

Universit Paris Diderot

On app que un a or t e de cours

Exerc ce 1 *Routage*

Le serveur S est connect la machine T par un r seau avec les noeuds A, B, C, D, les capacit s de connexions entre les noeuds sont en Mbit/s [illegible].

	A	B	C	D	T
S		[illegible]			
A		[illegible]			
B				[illegible]	[illegible]
C				[illegible]	
D	[illegible]				4

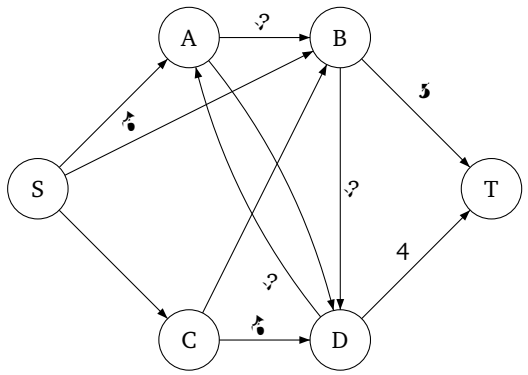
L'utilisateur de la machine T t l charge un tr s grand chier du serveur S. On veut trouver le routage qui maximise le d bit.

1. Quel probl me algorithmique de cours correspond ce probl me de routage ? Comment s'appelle l'algorithme du cours ?

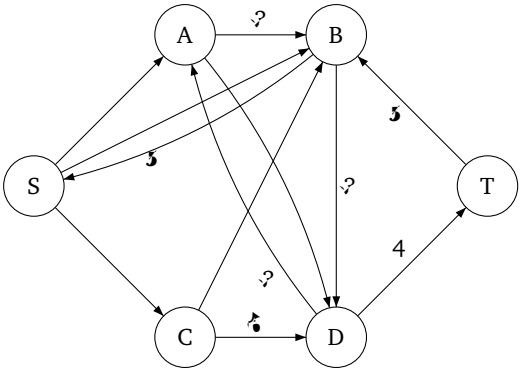
Correct on. Probl me de ot maximum dans un r seau. Algorithme de Ford-Fulkerson.

. Appliquez cet algorithme. Donnez toutes les tapes de son application.

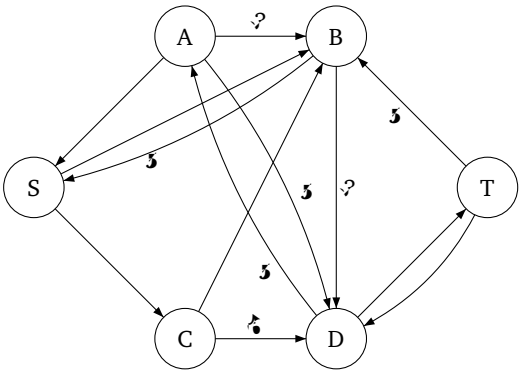
Correct on. Dessinons d'abord le r seau [illegible].



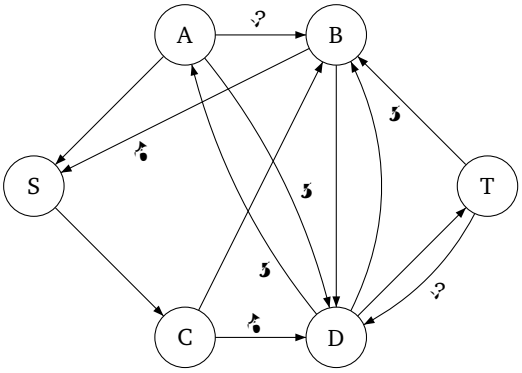
On trouve le premier chemin am liorant SBT de poids [illegible], et on dessine le r seau r siduel.



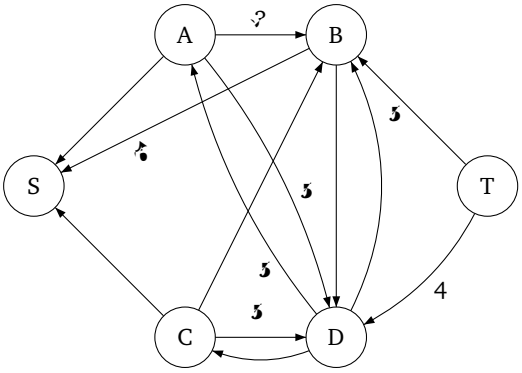
On trouve le second chemin am liorant SADT de poids , et on dessine le r seau r siduel.



On trouve le troisi me chemin am liorant SBDT de poids , et on dessine le r seau r siduel.



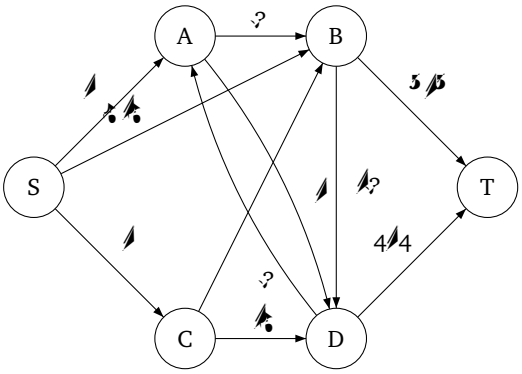
On trouve le quatri me chemin am liorant SCDT de poids , et on dessine le r seau r siduel.



Il ne reste plus de chemin am liorant, l'algorithme s'arr. te.
Le ot maximal est montr dans le point suivant

?. Quel est le d bit maximal, quel routage assure ce d bit ?

Correct on. Le d bit optimal est + + + , il est r alis par le routage suivant la somme de 4 ots sur SBT, sur SADT, sur SBDT, sur SCDT.



n a or t e fac e

Exerc ce | *Valeur encadrée*

tant donn un tableau tri d'entiers $A[s..f]$ et deux entiers "bornes" $a \leqslant b$, on cherche s'il existe un l ment $A[i]$ du tableau tel que $a \leqslant A[i] \leqslant b$ s'il y en a plusieurs trouvez un.

Exe p e Soit le tableau $A[1..5] = [3, 7, 8, 43, 556]$ et les bornes $a = 40, b = 50$. Dans ce cas-l la valeur encadr e existe c'est $A[4] = 43$.

1. Donnez en pseudocode un algorithme "diviser-pour-r gner" qui r sout ce probl me. Expliquez bri vement.

   Correct on. Coupons le tableau en deux moiti s $A[s..m]$ et $A[m+1..f]$. Les trois cas ci-dessous correspondent trois positions de l' l ment de milieu $A[m]$ par rapport l'intervalle $[a, b]$.

auc e: $A[m] < a$. Dans ce cas-l tous les l ments de la moiti gauche du tableau sont aussi $< a$, et ne peuvent pas appartenir l'intervalle demand $[a, b]$. On va chercher la valeur encadr e dans la moiti droite seulement.

Dedans: $a \leqslant A[m] \leqslant b$. On a d j trouv une valeur encadr e $A[m]$. On retourne son indice.

dro te: $b < A[m]$. Ce cas est sym trique au premier, la recherche peut , tre limit e la moiti gauche du tableau.

Cette analyse m ne la fonction r cursive suivante chercher s,f) qui renvoie l'indice de la valeur encadr e dans le tableau $A[s..f]$ ou $\perp$ s'il n'y en a pas. On suppose que le tableau A, et les bornes a,b sont des variables globales.

```
fonction chercher s,f)
        // cas de base
        si s=f)
                si A[s] ∈ [a; b])
                                retourner s
                sinon retourner ⊥

        // on divise pour régner
        m= s+f)/
        si A[m] < a
                ind= chercher m+ ,f)
        sinon si a ⩽ A[m] ⩽ b
                ind=m
        sinon
                ind=chercher s,m)

        retourner ind
```

. Analysez sa complexit .

Correct on. On a r duit un probl me de taille n un seul probl me de la taille $n/2$ et des petits calculs en $O(1)$, donc la complexit satisfait la r currence .

$$T(n) = T(n/2) + O(1).$$

En la r solvant par Master Theorem on obtient que $T(n) = O(\log n)$.

n eu - deux puzz es - deux a or t es

Un *Domino* est un rectangle qui contient lettres, par exemple | A | H | il ne peut pas , tre retourn).

è e: Une s quence de dominos est une *chaîne*, si les lettres voisines co.ncident sur chaque paire des dominos cons cutifs, par exemple | A | H | | H | K | | K | A | | A | Z | | Z | V | .

Donnée n t a e pour es deux puzz es: n pi ces de dominos D1 = | x1 | y1 | ,..., Dn = | xn | yn |

Puzz e 1 per utat on): Il faut trouver une permutation de tous ces dominos qui forme une cha.ne selon la r gle ci-dessus), si celle-l existe.

Puzz e sous-séquence): Il faut trouver la plus longue sous-s quence de tous ces dominos qui forme une cha.ne selon la r gle ci-dessus).

Exe p e: Pour les pi ces | A | H |, | A | Z |, | K | A |, | H | K |, | Z | V |.

- la solution du puzzle est | A | H | | H | K | | K | A | | A | Z | | Z | V |;
- la solution du puzzle est | A | H | | H | K |, ou bien | A | Z | | Z | V |.

Exerc ce 2 *Premier puzzle - backtracking*

. On cherche d velopper un algorithme de type backtracking qui r sout le puzzle .

1. crivez l'algorithme en pseudocode)

Correct on. On suppose que les pi ces sont repr sent es par deux tableaux x[1..n] et y[1..n]. On repr - sentera les solutions partielles du probl me par un tableau Chaine[1..k] qui contiendra les indices des pi ces de domino formant une cha.ne.
La fonction de backtracking recherche en profondeur) tant donn Chaine[1..k] cherche une solution de probl me qui commence par cette Chaine.

```
fonction chercher k)
        si k=n
                Af cher Chaine)
                stop

        pour tout ind∉ Chaine
                si k=0 || compatible Chaine[k] ,ind)
                  Chaine[k+1] =ind
                  chercher k+ )
```

La primitive compatible i,j) teste si on peut placer le d i gauche du d j.

```
fonction bool enne compatible i,j)
        return y[i] == x[j]
```

Le programme principal :

```
initialiser
chercher 0)
af cher "pas de solution"
```

. Expliquez cet algorithme vous pouvez vous inspirer de l'indication)

Correct on. On d nit une k-solution partielle comme une cha.ne compos e de k dominos parmi les n pi ces donn es). Une k-solution peut , tre tendue vers une k + 1-solution en ajoutant sa droite une pi ce qui) n'est pas encore utilis e et) est compatible. On fait une recherche en profondeur dans l'arbre se solutions partielles, en commençant par sa racine la 0-solution vide). On s'arr, te si on trouve une cha.ne qui utilise toutes s pi ces k=n).

3. Estimez le nombre d'op rations n cessaire

Correct on. Au pire cas on teste toutes les permutations, il y en a $n!$ ce qui donne une complexit exponentielle.

4. Montrez le d roulement de votre algorithme pour les pi ces
[A | H], [A | Z], [K | A], [H | K], [Z | V].

Correct on.
- vide
- [A | H]
- [A | H] [H | K]
- [A | H] [H | K] [K | A]
- [A | H] [H | K] [K | A] [A | Z]
- [A | H] [H | K] [K | A] [A | Z] [Z | V]

L'algorithme s'est donc d roul sur une seule branche de l'arbre, sans retour arri re.

Indication : Essayez de répondre aux questions suivantes pour parvenir à un tel algorithme.
- Comment tester que deux dominos peuvent être adjacents ?
- Comment définir une solution partielle ?
- Quelle est une solution partielle de taille 0 ?
- Comment passer d'une solution partielle à une solution plus grande ?
- Quand s'arrêter ?
- Comment représenter une solution partielle par une structure de données ? Comment représenter les dominos déjà utilisés en chaîne et les dominos encore disponibles ?

Exerc ce 4 *Deuxième puzzle - programmation dynamique*

On cherche d velopper un algorithme de type programmation dynamique qui r sout le puzzle .

1. Soit $c(i)$ une fonction enti re qui donne la longueur de la cha ne la plus longue qui est une sous-s quence de $D_1, D_2, \ldots D_i$ et qui se termine par la pi ce D_i. crivez les quations de r currence pour cette fonction sans oublier les cas de base.

Correct on. Cas de base $c(1) = 1$. Pour $i > 1$ la sous-cha ne qui nous int resse est la plus longue parmi les cha nes suivantes :
- Le d D_i seul
- Pour un $j < i$ et tel que D_j est compatible avec D_i on prend la cha ne la plus longue jusqu' D_j, et on y ajoute le dernier d D_i.

a donne la r currence suivante

$$c(i) = \begin{cases} 1, & \text{si } i = 1 \\ \max\{1, c(j) + 1 \mid & j < i \text{ tel que compatible}(j, i)\} \text{si } i > 1 \end{cases}$$

. crivez un algorithme ef cace r cursif avec "marquage" ou it ratif pour calculer c .

Correct on. L'algo it ratif remplit le tableau $\overline{d}\,\overline{i}$ par les valeurs de la fonction c i pour i de 1 jusqu' n.

```
d =
pour i de'    n faire
    meilleur=
    pour j de    i-  faire
        si compatible j,i et d j+ > meilleur
            meilleur=d j +
    d i =meilleur
```

2. En sachant calculer la fonction choisie c, comment trouver la longueur de la sous-cha ne la plus longue parmi $D_1, D_2, \ldots D_i$.

Correct on. Le r sultat nal est l' l ment maximum de tout le tableau $\overline{d}$ ..n

4. Analysez la complexit de votre algorithme.

Correct on. $O(n^2)$ cause de deux boucles imbriqu es.

. Montrez le d roulement de votre algorithme pour les pi ces
$\boxed{A \mid H}$, $\boxed{A \mid Z}$, $\boxed{K \mid A}$, $\boxed{H \mid K}$, $\boxed{Z \mid V}$.

Correct on.
- $C[1] = 1$
- $C[2] = 1$ il n'y a pas de d compatible gauche de la - me pi ce).
- $C[3] = 1$ m. me raison).
- $C[4] = \max\{1, C[1] + 1\} = 2$ les d s et 4 sont compatibles).
- $C[5] = \max\{1, C[2] + 1\} = 2$ les d s et 5 sont compatibles).

6. Comment modi er votre algorithme pour qu'il trouve la sous-cha ne la plus longue et non seulement sa longueur).

Correct on. Pour chaque i il faut m moriser le meilleur d j mettre gauche de D_i. a donne

```
D[1] = J[1] = 1
pour i de 2 n faire
    meilleur=1
    pour j de 1 i-1 faire
        si compatible(j,i) et D[j]+1 > meilleur
            meilleur=D[j] + 1 J[i] =j
    D[i] =meilleur
```

On prend le i qui maximise C[i]