Algorithmique avancée : Contrôle Continu n^{o}1

Vendredi 21 Octobre 2016. Durée 2h. Documents autorisés.

Le barème est donné à titre indicatif. Toutes les réponses doivent être soigneusement **justifiées**. La compréhension du sujet faisant partie de l'épreuve, **on ne répondra à aucune question**. Si vous rencontrez des ambiguïtés, vous expliquerez **sur votre copie** comment vous les interprétez.

I Complexité (4 points)

1. Un algorithme a une complexité temporelle qui peut s'exprimer sous la forme suivante :

$$T(n) = 27T(n/3) + 7n^3 + 4$$

Donnez une estimation asymptotique Θ de cette complexité.

Solution: (2 points) $n^{\log_3 27} = n^3$, $f(n) = \Theta(n^3)$. donc Master theorem avec Cas 2 : $T(n) \in \Theta(n^3 \log n)$.

2. On cherche à avoir une approximation de $\log(n!)$. Pour cela, on considère l'approximation de Stirling : $n! \sim \sqrt{2.\pi} n^{n+\frac{1}{2}} e^{-n}$ et on note :

$$f(n) = n^{n+\frac{1}{2}} e^{-n}$$

donnez une approximation asymptotique en Θ de la fonction $\log(f(n))$ puis conclure sur $\log(n!)$.

Solution: (2 points) $\log(f(n)) = (n + \frac{1}{2}) \log n - n \log e = n \log n + \frac{1}{2} \log n - n \log e$

- $\log(f(n)) \leq n \log n + n \log n - n \log e$ quand $n \geq \frac{1}{2}$
  or $n \log n - n \log e = n(\log n - \log e) \leq n \log n$ quand $n > 1$ car $\log e \geq 0$ ($\log e = 0.4343$)
  donc $\log(f(n)) \leq 2n \log n$ et $\log(f(n)) \in O(n \log n)$
- $\log(f(n)) = n \log n + \frac{1}{2} \log n - n \log e$ or $\log e < 0.5$ donc $n \log e < \frac{1}{2} n$ et ainsi $\log(f(n)) \geq \frac{1}{2} n \log n + \frac{1}{2}(n \log n + \log n - n)$ ce qui signifie que $\log(f(n)) \geq \frac{1}{2}(n \log n)$ si $n > 10$ (puisque $n \log n + \log n - n \geq 0$ quand $\log n > 1$ ce qui est vrai si $n > 10$).
  Ainsi $\log(f(n)) \in \Omega(n \log n)$

Donc $\log(f(n)) \in \Theta(n \log n)$. puisque $\log \sqrt{2\Pi}$ est une constante positive $\log \sqrt{2\Pi} + \log(f(n)) \geq \frac{1}{2}(n \log n)$ et $\log \sqrt{2\Pi} + \log(f(n)) \leq 3(n \log n)$ si $n \log n \geq \log \sqrt{2\Pi}$ ce qui est vrai quand $n \geq 2\Pi$. Donc $\log \sqrt{2\Pi} f(n) \in \Theta(n \log n)$.

(Partie considérée en Bonus) On admet sans preuves les deux propriétés suivantes :

1. *Pour toutes fonctions f et g à valeurs strictement positives, $f \sim g$ et $\lim_{n \to +\infty} f(n) \neq 1$ implique $\log f \sim \log g$*
2. *Pour toutes fonctions f, g et h, $f \sim g$ et $f(n) \in \Theta(h(n))$ implique $g(n) \in \Theta(h(n))$.*

Donc $\log n! \in \Theta(n \log n)$.

II Structures de données (8 points)

On considère le tableau $T1$ contenant les 19 éléments suivants :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
10	36	9	12	25	8	14	29	18	11	17	38	27	3	6	89	13	77	21

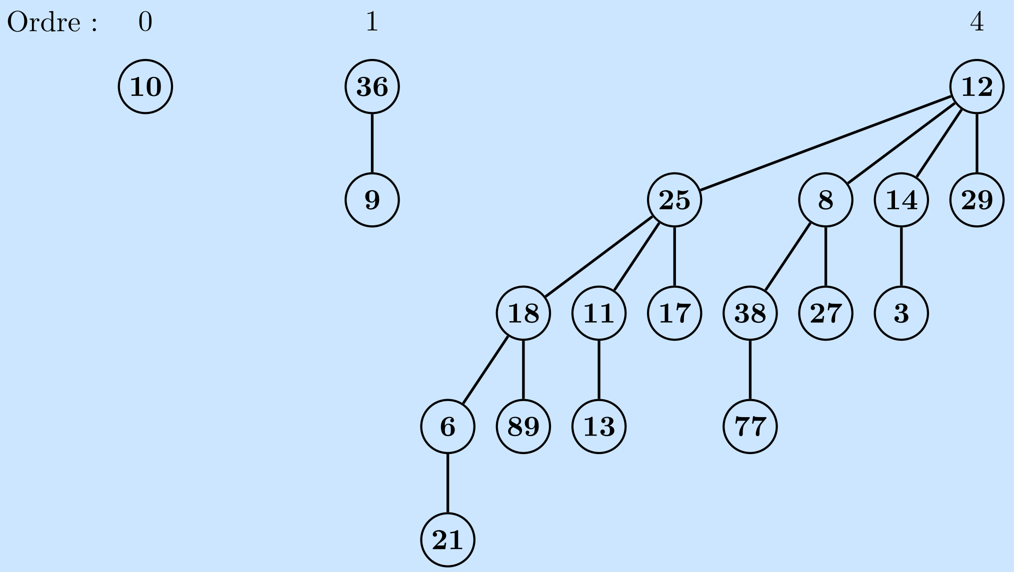
1. On désire créer un tas binomial à partir du tableau initial $T1$,

   a) Donnez le nombre d'arbres binomiaux avec leurs ordres pour un tas binomial contenant le nombre d'éléments du tableau $T1$.

   Solution: (0.5 point) 19=16+2+1. donc un arbre d'ordre 0, un d'ordre 1, un d'ordre 4

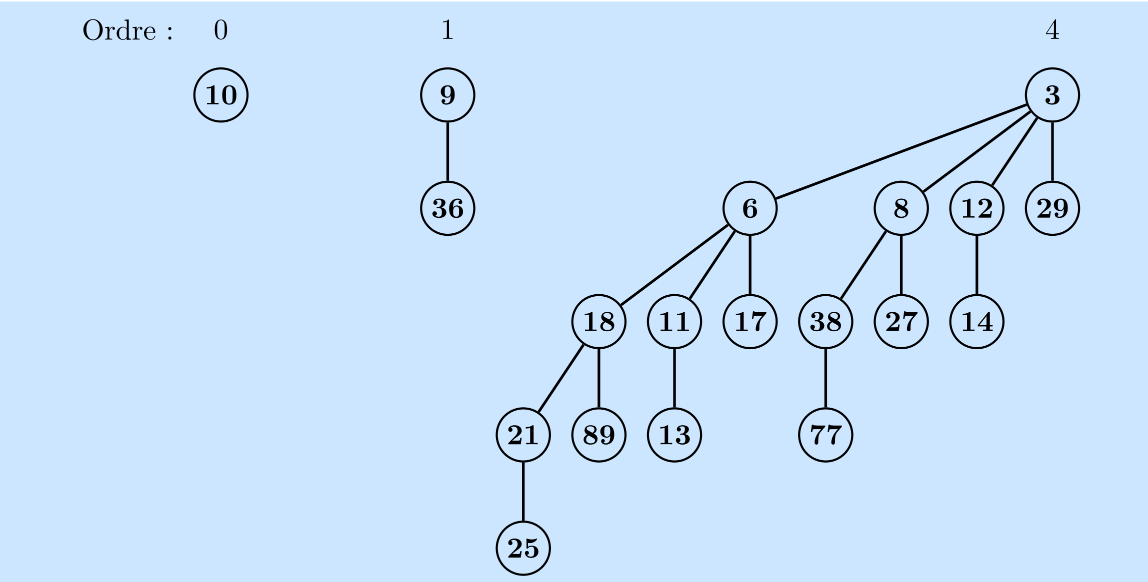
   b) Créez la forêt $F1$ des arbres binomiaux décrits à la question précédente en remplissant les noeuds des arbres avec les éléments du tableau initial $T1$. Vous positionnerez les éléments au premier noeud vide du premier arbre binomial non plein en remplissant chaque arbre de gauche à droite et de haut en bas.

   Solution: (0.5 point)



2. Cette forêt $F1$ n'est pas un tas binomial, quelles opérations faut-il faire pour la transformer en tas binomial ? Effectuez ces opérations et décrivez le tas binomial obtenu $F2$.

   Solution: (3 points) percolate down depuis les noeuds internes à partir des noeuds internes de prof max jusqu'à ceux de prof 0 (la racine).



3. Combien de feuilles y a-t-il dans un arbre binomial d'ordre k ?

Solution: (1 point) ordre 0 : 1 feuille, ordre 1 : 1 feuille, ordre 2 : deux feuilles. arbre d'ordre 3= 2 arbres d'ordre 2= 4 feuilles. Feuilles(arbre d'ordre k)= 2.Feuilles(arbre d'ordre k-1). On montre par récurrence que Feuilles(arbre d'ordre k)= 2^{k-1}.

4. Quelle est la complexité en pire cas de toute la procédure de création de tas binomial que l'on vient d'effectuer en fonction de la taille initiale n du tableau ? (vous pouvez donner une expression en fonction du nombre de Fibonacci)

Solution: (1 point) création des n noeuds $O(n)$ puis percolate dans chaque arbre binomial, nombre de feuille d'un arbre binomial d'ordre $k= 2^{k-1}$, donc percolate down de $2^k - 2^{k-1} = 2^{k-1}$ noeuds au pire de toute la hauteur $= k$ donc pour un arbre ça donne $O(k.2^{k-1})$ ceci pour $k = 1$ à $\log_2 n$, donc $O(n + \sum_{k=1}^{[\log_2 n]} k.2^{k-1})$. (cette expression est celle qu'on attendait des étudiants, on peut s'en contenter car l'utilisation du nombre de Fibonacci est moins évidente...) Notons que l'expression $\sum_{k=1}^{[\log_2 n]} k.2^{k-1}$ peut se réduire ensuite à $1 + (\log_2 n - 1).2^{\log_2 n} = 1 + n(\log_2 n - 1)$ (en utilisant l'égalité $1 + 2x + 3x^2 + \cdots + nx^{n-1} = (1-(n+1)x^n + nx^{n+1})/(1-x)^2$.) ce qui donne au final $O(n \log n)$.

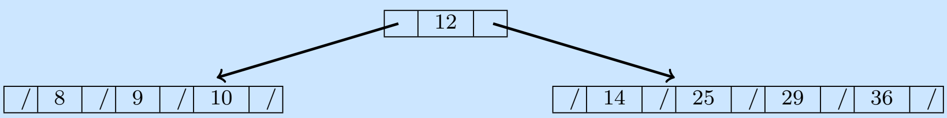
5. Dessinez le B-Arbre (B-Tree en anglais) de degré minimum $t = 4$ résultant de l'insertion successive des nombres du tableau $T1$. Vous dessinerez également les arbres intermédiaires avant et après chaque éclatement de noeuds.

Solution: (2 points)

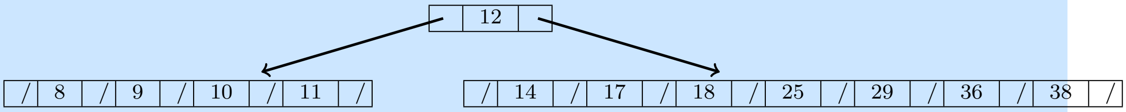
1.

/	8	/	9	/	10	/	12	/	14	/	25	/	36	/

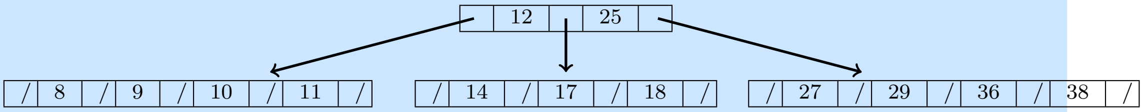
2. insertion 29 (8ème élément)



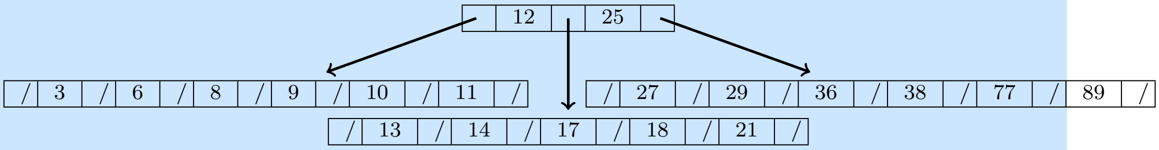
3.



4. insertion du 27 (13ème élément)



5. dernier

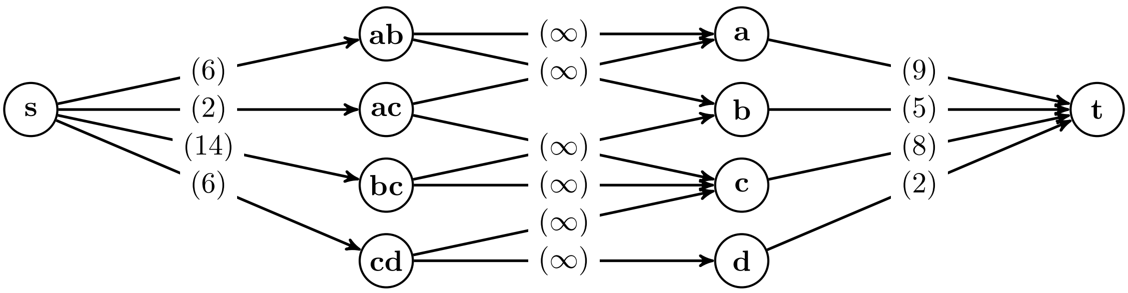


III Le problème de sélection (8 points)

On dispose d'un ensemble O de 4 objets a, b, c et d, chacun a un coût, leurs coûts respectifs sont 9, 5, 8 et 2. Certaines combinaisons d'objets sont plus ou moins utiles, on considère donc que certains ensembles rapportent un gain, soit E l'ensemble des combinaisons qui rapportent. Ici, $\{a, b\}$ rapporte 6, $\{a, c\}$ rapporte 2, $\{b, c\}$ rapporte 14, $\{c, d\}$ rapporte 6.

L'objectif est de sélectionner un ensemble d'objets Y de façon à obtenir un bénéfice maximum (c'est-à-dire que la différence entre la somme des gains des combinaisons que forment les objets de Y et la somme des coûts des objets de Y, soit maximale).

On représente ce problème par le réseau suivant :



1. a) Donnez un exemple non trivial de coupe de capacité finie (c'est-à-dire différent de $(\{s\}, X \setminus \{s\})$ et de $(X \setminus \{t\}, t)$ avec sa capacité.

   Solution: (1 point) $(\{s, ab, a, b\}, \{ac, bc, cd, c, d, t\})$ capa=9+5+2+14+6=36

   b) Les coupes de capacité finies correspondent à la sélection de certains objets, donnez pour votre coupe les objets sélectionnés. Calculez le gain et le coût associé à cette sélection.

   Solution: (0.5 point) a et b, coût 9+5 ; gain= 6 ;

2. Calculez une coupe de capacité minimum sur ce réseau. Vous détaillerez les étapes du calcul et la méthode utilisée. Listez ses arcs.

   Solution: (3 points) coupe capa min= flot max donc algo Ford Fulkerson. On part d'un flot nul. Chaînes augmentantes :

— (s bc b t) : on augmente de 5 sur le cycle (s bc b t s).
— (s bc c t) : on augmente de 8 sur le cycle (s bc c t s).
— (s ab a t s) : on augmente de 6 sur le cycle (s ab a t s).
— (s ac a t s) : on augmente de 2 sur (s ac a t s).
— (s cd d t s) : on augmente de 2 sur (s cd d t s)

on ne peut marquer que s, bc, cd, b, c, d donc Flot max. valeur 23. donc coupe capa min=23 = ({s,bc,cd,b, c, d},{ab, ac, a, t}). arcs=(s,ab)(s,ac),(b,t)(c,t),(d,t)

3. Soit A un ensemble de sommets tel que $(A, X \setminus A)$ est une coupe de capacité finie.

a) Montrez que $\forall u \in \omega^+(A)$, on a $u = (s, e)$ ou bien $u = (o, t)$ avec e un sommet représentant une combinaison d'objets ($e \in E$) et o un objet ($o \in O$).

Solution: (1 point) si la coupe est de capacité finie alors elle ne contient aucun arc de E vers O, donc les seuls arcs u qu'elle peut contenir sont des arcs $u = (s, e)$ ou $u = (o, t)$.

b) Soit une coupe de capacité finie $C = (A, X \setminus A)$, et soit Y l'ensemble de tous les objets apparaissant dans les arcs d'extrémité t dans cette coupe. Montrez que

$$capa(C) = \sum_{o \in Y} cout(o) + \sum_{e \in E_{\overline{Y}}} gain(e)$$

où $E_{\overline{Y}} = \{e \in E \text{ et } e \not\subseteq Y\}$ est l'ensemble des sommets représentant des combinaisons dont les objets ne sont pas tous dans Y.

Solution: (1 point) D'après la question précédente,

$$capa(C) = \sum_{u=(o,t) \in \omega^+(A)} capa(u) + \sum_{u=(s,e) \in \omega^+(A)} capa(u)$$

Pour chaque objet o apparaissant dans un arc (o, t) de la coupe, la capacité de l'arc $(o, t) = cout(o)$, d'après le dessin proposé, chaque ensemble d'objet e est relié à tous les objets qu'il contient. si un objet o d'un ensemble e n'est pas dans Y alors e n'est pas dans A car sinon la coupe aurait un arc de e vers o (donc de capacité infinie). si e n'est pas dans A alors $(s, e) \in \omega^+(A)$ (puisque par définition s est dans A). L'ensemble des e hors de A est noté $E_{\overline{Y}}$ il correspond aux ensembles dont au moins un objet n'est pas dans Y. On retrouve donc la formule proposée car la capacité d'un arc (s, e) est égale au gain de l'ensemble e.

c) En déduire que

$$capa(C) = \sum_{o \in Y} cout(o) + \sum_{e \in E} gain(e) - \sum_{e \in E, e \subseteq Y} gain(e)$$

et que l'on peut réécrire cette égalité en

$$capa(C) = \sum_{e \in E} gain(e) - gainNet(Y)$$

vous direz ce que représente $gainNet(Y)$.

Solution: (0.5 point) Puisque $E_{\overline{Y}} = \{e \in E$ et $e \not\subseteq Y\}$ alors $E_{\overline{Y}} = E \setminus \{e \in E$ et $e \subseteq Y\}$ d'où $\sum_{e \in E_{\overline{Y}}} gain(e) = \sum_{e \in E} gain(e) - \sum_{e \in E, e \subseteq Y} gain(e)$. Donc $gainNet(Y) = \sum_{e \in E, e \subseteq Y} gain(e) - \sum_{o \in Y} cout(o)$ c'est ce que rapporte l'ensemble Y gain des combinaisons qu'on peut réaliser avec les objets de Y moins le coût des objets de y.

d) En déduire que maximiser ce que rapporte une sélection d'objet Y revient à sélectionner une coupe de capacité minimum. Quels objets faut-il sélectionner ? Pour quel bénéfice ?

Solution: (1 point) Maximiser le gainNet(Y) diminue la capacité de la coupe C puisque le gainNet apparait avec un coefficient négatif et que la somme des gain de tous les ensembles est une constante indépendante des objets sélectionnés. Donc trouver une coupe min est équivalent à sélectionner un semble d'objet qui rapporte un bénéfice net maximum.

Ici la coupe min de 23 correspond à Y=$\{b, c, d\}$ il faut donc sélectionner les objets b, c et d qui coûtent 5+8+2=15 mais qui permettent de réaliser les combinaisons bc et cd qui rapportent 14 + 6=20 on a donc un bénéfice de 5 et c'est le meilleur possible.