Outils logiques et algorithmiques – Partiel – Mars 2023

Durée 2h. Le sujet comporte 4 exercices indépendants. Notes personnelles et documents de cours autorisés.
Barème approximativement proportionnel au temps nécessaire à la résolution de chaque question.

Exercice 1. Petit calcul *(10 minutes)*

On note I_k la somme des nombres impairs compris entre 0 et k. Par exemple : $I_5 = 1 + 3 + 5 = 9$. Montrer que

$$\forall n \in \mathbb{N},\ I_{2n} = n^2$$

Technique au choix : récurrence ou calcul astucieux.

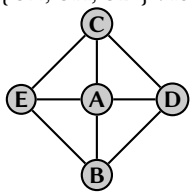
Correction. On note $P(n)$ la propriété « $I_{2n} = n^2$ ». Par récurrence :

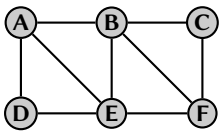
- Cas de base ($P(0)$) : $I_{2n} = 0 = 0^2$.
  Note : une somme de zéro éléments vaut par convention 0 (car ce nombre est l'élément neutre de l'opération d'addition).
- Hérédité : soit n tel que $I_{2n} = n^2$. Alors $I_{2(n+1)} = I_{2n+2} = I_{2n} + 2n + 1 = n^2 + 2n + 1 = (n+1)^2$.

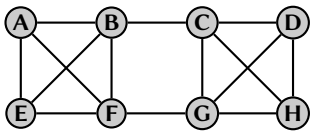
Exercice 2. Coupures dans un graphe *(50 minutes)*

On appelle *coupure* d'un graphe G non orienté connexe avec au moins deux sommets, un ensemble C d'arêtes tel que le graphe G sans les arêtes de C n'est plus connexe (il existe deux sommets qui ne sont pas reliés par un chemin). On dit que la coupure est *minimale* s'il n'y a pas d'autre coupure avec moins d'arête.

On utilisera comme exemples les graphes G_1, G_2 et G_3 suivants. Pour déconnecter G_1 il faut retirer au moins 3 arêtes (par exemple $\{CE, CA, CD\}$: le sommet C est alors isolé du reste du graphe).

G_1 

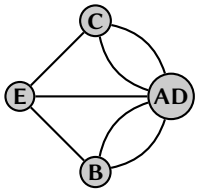
G_2 

G_3 

Questions.

1. Donner une coupure minimale pour chacun des graphes G_2 et G_3.
2. Que peut-on dire de la taille de la coupure minimale par rapport au plus petit degré d'un sommet d'un graphe G quelconque ?
3. Donner un exemple de graphe connexe dont tous les sommets ont un degré plus grand que deux mais qui a une coupure de taille 1.
4. Si le graphe n'a que deux sommets, quelle est la coupure minimale ?
5. Une boucle peut-elle appartenir à une coupure minimale ? pourquoi ?

Soit un graphe G et une arête a de ce graphe entre les sommets s et s'. On construit un graphe G' en retirant l'arête a et en fusionnant les deux sommets s et s' en un nouveau sommet que l'on notera ss'. Toutes les arêtes qui arrivaient à s ou s' dans G arrivent dans G' sur le sommet ss'. Les autres extrémités ne sont pas changées. On peut créer par ce processus des arêtes parallèles (mais on supprime les boucles qui pourraient être ainsi créées par les arêtes entre s et s'). Sur l'exemple du graphe G_1, en retirant ainsi l'arête entre les sommets A et D on obtient le graphe ci-contre.



Questions.

6. Dessiner les graphes obtenus par ce procédé à partir du graphe G_3 en fusionnant d'abord les sommets E et F puis, en fusionnant les sommets C et G dans le graphe obtenu à l'étape précédente.
7. Dans le graphe obtenu à la question précédente, est-il possible de fusionner les sommets EF et CG ?
8. Comparer dans le cas général le nombre de sommets et le nombre d'arêtes dans G et G'.
9. Comparer le degré d'un sommet dans le graphe G et le degré du sommet correspondant dans le graphe G'. Traiter le cas d'un sommet qui n'est ni s ni s' puis le cas du sommet fusionné ss'.
10. Montrer que s'il y a un chemin entre deux sommets du graphe G' alors il y a un chemin entre les sommets correspondants dans le graphe G. Si le chemin a pour extrémité le sommet fusionné ss' dans G' on montrera qu'il y a dans G un chemin d'extrémité s et un d'extrémité s'.
11. En déduire qu'une coupure de G' est aussi une coupure de G.
12. Si la coupure est minimale dans G', est-elle aussi minimale dans G ?

Remarque. Ce résultat est à la base d'un algorithme probabiliste de recherche de coupure minimale dans un graphe : on choisit aléatoirement des arêtes que l'on supprime au fur et à mesure par la méthode précédente jusqu'à obtenir seulement deux sommets pour lesquels la coupure est évidente. On peut itérer le processus plusieurs fois pour trouver des solutions suffisament "petites".

Correction.

1. Pour G_2, il suffit de déconnecter un sommet de degré 2, par exemple D, une coupure minimale est $\{AD, DE\}$.
   Pour G_3, les sommets ont tous un degré au moins 3, mais on remarque qu'en enlevant les arêtes $\{BC, FG\}$, il n'y a plus de chemin possible de B à C.
2. Si on retire toutes les arêtes qui arrivent sur un sommet alors il n'y a plus de chemin de ce sommet au reste du graphe, c'est donc une coupure. Ce n'est pas forcément la plus petite. La coupure minimale a donc en général une taille inférieure ou égale au plus petit degré des sommets du graphe.
3. Il suffit de prendre un graphe formé de deux triangles et de relier ces deux triangles par une arête. Tous les sommets ont un degré au moins 2, mais on obtient une coupure en retirant seulement une arête (celle qui relie les deux triangles).
4. Si le graphe n'a que deux sommets s et s' alors la coupure minimale est l'ensemble des arêtes de s à s'.
5. S'il y a un chemin dans un graphe entre deux points, alors il y a un chemin sans boucle. Donc si une arête a qui est une boucle appartient à une coupure C alors $C\backslash\{a\}$ est aussi une coupure du graphe (les deux points qui étaient déconnectés le reste en ajoutant a). Et donc la coupure C n'est pas minimale.
6.
7.
8. G' a un sommet de moins que G, il a strictement moins d'arêtes (plus précisément, le nombre d'arêtes de G' est égal au nombre d'arêtes de G moins le nombre d'arêtes dans G de s à s').
9. Le degré des sommets qui sont à la fois dans G et G' ne change pas. Le degré dans G de ss' est égal au degré de s dans G plus le degré de s' dans G moins le nombre d'arêtes dans G de s à s'.
10. Supposons qu'il y a un chemin dans G' entre les sommest t et t'. On peut sans perdre de généralité supposer que ce chemin ne passe pas deux fois par le même sommet. Si ce chemin ne passe pas par le sommet ss' alors c'est aussi un chemin dans G. Si ce chemin passe par le sommet ss', matérialisons les arêtes arrivant à et repartant de ce sommet : $\xrightarrow{b} ss' \xrightarrow{c}$. Par cas sur la cible de $\xrightarrow{b}$ et la source de $\xrightarrow{c}$ dans G (qui sont soit s soit s') :
    - s'il s'agit du même sommet, alors on a le chemin $\xrightarrow{b} s \xrightarrow{c}$ ou $\xrightarrow{b} s' \xrightarrow{c}$ dans G,
    - si l'une est s et l'autre est s', alors on a le chemin $\xrightarrow{b} s \to s' \xrightarrow{c}$ ou $\xrightarrow{b} s' \to s \xrightarrow{c}$ dans G.

    Le cas où le sommet ss' est une extrémité est similaire. Considérons le cas où c'est la fin du chemin. Si $\xrightarrow{b} s$ dans G alors on a un chemin dans G qui arrive jusqu'à s, et avec $\xrightarrow{b} s \to s'$ on en construit un toujours dans G qui arrive à s'.
11. (Question rendue optionnelle car mal formulée... ce n'est pas une conséquence du résultat précédent) On montre la contraposée de la question précédente. S'il y a un chemin dans G alors il y a aussi un chemin dans G'. Si le chemin passe par une arête $s \xrightarrow{a} s'$ (qui disparait dans G') alors il suffit de la retirer : on remplace $\to s \xrightarrow{a} s' \to$ dans G par le sommet $\to ss' \to$ dans G'. Sinon toutes les arêtes du chemin dans G sont aussi des arêtes de G', si le chemin dans G passait par s ou s' il suffit de remplacer le sommet par ss' dans le chemin qui devient ainsi un chemin de G'.
    Si C est une coupure dans G' alors C est aussi un ensemble d'arêtes de G. Il y a deux sommets t et t' tel qu'il n'existe pas de chemin de t à t' dans G'. S'il y avait un chemin de t à t' dans G alors on vient de montrer qu'il y en aurait aussi un dans G' ce qui contredit l'hypothèse de coupure. Donc C est aussi une coupure du graphe G.
12. Il peut exister des coupures plus petites dans G que dans G'. Si on prend par exemple le graphe de la question 3 et que l'on retire du graphe justement l'arête formant la coupure minimale, on obtient un graphe dont les coupures minimales ont une taille 2.

Exercice 3. Plus long préfixe commun *(20 minutes)*

La fonction suivante renvoie le plus long préfixe commun à un ensemble `t` de chaînes de caractères, c'est-à-dire une chaîne `r` telle que toutes les chaînes de `t` commencent par `r`.

```
1  static String commonPrefix(String[] t) { return commonPrefix(t, 0, t.length()); }
2
3  static String commonPrefix(String[] t, int lo, int hi) {
4    if (hi - lo <= 1) { return t[lo]; }
```

```
5     int mid = lo + (hi-lo)/2;
6     String s1 = commonPrefix(t, lo, mid);
7     String s2 = commonPrefix(t, mid, hi);
8     return commonP(s1, s2);
9   }
10
11  static String commonP(String s1, String s2) {
12    int n = Math.min(s1.length(), s2.length());
13    int i = 0;
14    while (i < n && s1.charAt(i) == s2.charAt(i)) { i++; }
15    return s1.substring(0, i);
16  }
```

Questions.

1. Détailler l'exécution de la fonction sur le tableau `{ "barbe", "barbare", "babar", "baba" }`.
2. Quelle est la complexité de `commonP` dans le meilleur cas ? dans le pire cas ? (compter les comparaisons de caractères)
3. Donner des équations récursives pour la complexité de `commonPrefix` dans le meilleur cas, et les résoudre.
4. Donner des équations récursives pour la complexité de `commonPrefix` dans le pire cas, et les résoudre.

Correction.

1. Appel principal : `commonPrefix(t, 0, 4)` déclenche deux appels récursifs.
   - `commonPrefix(t, 0, 2)` déclenche 2 appels.
     - `commonPrefix(t, 0, 1)`, renvoie `"barbe"`.
     - `commonPrefix(t, 1, 2)`, renvoie `"barbare"`.

     Préfixe commun renvoyé : `"barb"`.
   - `commonPrefix(t, 2, 4)` déclenche 2 appels.
     - `commonPrefix(t, 2, 3)`, renvoie `"babar"`.
     - `commonPrefix(t, 3, 4)`, renvoie `"baba"`.

     Préfixe commun renvoyé : `"baba"`.

   Préfixe commun renvoyé : `"ba"`.
2. Meilleur cas : temps constant (arrêt après la première comparaison). Pire cas : cas proportionnel à la longueur des mots (deux mots identiques).
3. On note $C_m(n)$ le nombre de comparaisons de caractères pour un tableau de n chaînes de caractères, dans le meilleur cas.

   $$\begin{aligned} C_m(1) &= 0 \\ C_m(n) &= 1 + C_m(\lfloor n/2 \rfloor) + C_m(\lceil n/2 \rceil) \qquad n > 1 \end{aligned}$$

   Résolution dans le cas où n est une puissance de 2 :

   $$\begin{aligned} C_m(2^0) &= 0 \\ C_m(2^k) &= 1 + C_m(\lfloor 2^k/2 \rfloor) + C_m(\lceil 2^k/2 \rceil) = 1 + 2C_m(2^{k-1}) \qquad k > 0 \end{aligned}$$

   Si on divise des deux côtés par 2^k on obtient

   $$\frac{C_m(2^k)}{2^k} = \frac{1}{2^k} + \frac{C_m(2^{k-1})}{2^{k-1}} = \sum_{0<j\leq k} \frac{1}{2^j} + \frac{C_m(2^0)}{2^0} = \sum_{0<j\leq k} \frac{1}{2^j} = 1 - \frac{1}{2^k}$$

   D'où finalement, en remultipliant par 2^k des deux côtés

   $$C_m(2^k) = 2^k - 1$$

   Complexité linéaire en le nombre de chaînes du tableau.
4. On note $C_p(n, s)$ le nombre de comparaisons de caractères dans le pire cas pour un tableau de n chaînes de caractères, chaque chaîne étant de longueur s.

   $$\begin{aligned} C_p(1, s) &= 0 \\ C_p(n, s) &= s + C_p(\lfloor n/2 \rfloor, s) + C_p(\lceil n/2 \rceil, s) \qquad n > 1 \end{aligned}$$

   La résolution est identifique, et aboutit à

   $$C_p(2^k, s) = s \times (2^k - 1)$$

   Complexité proportionnelle au produit du nombre de chaînes et de leur longueur.

Exercice 4. Plus long palindrome dans un texte *(40 minutes)*

La fonction suivante détermine la longueur du plus long palindrome contenu dans la chaîne `s`. *Rappel : un palindrome est une chaîne qui est identique lue de gauche à droite ou de droite à gauche, comme* `"1001001"`.

```
1  static int longestPalindrome(String s) {
2    int n = s.length();
3    int lMax = 0;
4    int a = 0, b = 0;
5    while (b < n) {
6      int r = 0; // ``rayon'' du palindrome
7      while (a-r >= 0 && b+r < n && s.charAt(a-r) == s.charAt(b+r)) { r++; }
8      int l = 2*r + b-a - 1; // longueur du palindrome
9      if (l > lMax) { lMax = l; }
10     if (a == b) { b++; }  // passage à la prochaine position
11     else        { a++; }
12   }
13   return lMax;
14 }
```

Questions.

1. Détailler l'exécution de cette fonction sur l'entrée `"babba"` sous la forme d'un tableau donnant, par ordre chronologique :
   - la valeur de la variable `r` et le résultat du test ligne 7 à chaque exécution de cette ligne,
   - les valeurs des variables `a`, `b` et `lMax`, et le résultat du test ligne 10 à chaque tour de la boucle externe.
2. Donner un invariant de la boucle externe caractérisant la relation entre `a` et `b`.
3. Donner un invariant de la boucle interne, qui précise autant que possible la présence d'un palindrome « autour » des indices `a` et `b`.
4. Donner un variant justifiant la terminaison de la boucle externe, et préciser le nombre de tours effectués.
5. Donner un variant justifiant la terminaison de la boucle interne, et donner un encadrement du nombre de comparaisons de caractères effectuées pour une exécution de cette boucle. Donner des exemples pour lesquels les bornes de cet encadrement sont atteintes.
6. Sur la base des questions précédentes, donner un encadrement du nombre de comparaisons de caractères effectuées par `longestPalindrome`, en fonction de la longueur n de la chaîne donnée en paramètre.
   On ne cherchera pas à savoir s'il existe des combinaisons de lettres permettant d'atteindre les bornes de cet encadrement.

Correction.

1.

Ligne	a	b	lM	r	test
4	0	0	0	–	
5	0	0	0	–	$0 < 5$
7	0	0	0	0	$0 \geq 0 \wedge 0 < 5 \wedge b = b$
7	0	0	0	1	$-1 \not\geq 0$
10	0	0	0	–	$0 = 0$
5	0	1	0	–	$1 < 5$
7	0	1	0	0	$b \neq a$
10	0	1	0	–	$0 \neq 1$
5	1	1	0	–	$1 < 5$
7	1	1	0	0	$1 \geq 0 \wedge 1 < 5 \wedge a = a$
7	1	1	0	1	$0 \geq 0 \wedge 2 < 5 \wedge b = b$
7	1	1	0	2	$-1 \not\geq 0$
9	1	1	0	2	$3 > 0$
10	1	1	3	–	$1 = 1$
5	1	2	3	–	$2 < 5$
7	1	2	3	0	$a \neq b$
10	1	2	3	–	$1 \neq 2$
5	2	2	3	–	$2 < 5$
7	2	2	3	0	$2 \geq 0 \wedge 2 < 5 \wedge b = b$
7	2	2	3	1	$a \neq b$
10	2	2	3	–	$2 = 2$
5	2	3	3	–	$3 < 5$
7	2	3	3	0	$2 \geq 0 \wedge 3 < 5 \wedge b = b$
7	2	3	3	1	$1 \geq 0 \wedge 4 < 5 \wedge a = a$
7	2	3	3	2	$5 \not< 5$
9	2	3	3	2	$4 > 3$
10	2	3	4	–	$2 \neq 3$
5	3	3	4		...
5	3	4	4		...
5	4	4	4		...
5	4	5	4		fin

2. Invariant : $b = a \vee b = a + 1$ (a et b sont égaux, ou b est supérieur de 1 à a). Invariant assuré par les lignes 11 et 12 qui incrémentent l'un ou l'autre.
3. Invariant : $\forall i \in [0, r[\,, s[a-i] = s[b+i]$ (les r caractères lus à partir de a vers la gauche sont égaux aux r caractères lus à partir de b vers la droite). Autrement dit, soit a et b sont égaux et $s[a - r + 1, a + r - 1]$ est un palindrome de longueur $2r - 1$, soit $b = a + 1$ et $s[a - r + 1, b + r - 1] = s[a - r + 1, a + r]$ est un palindrome de longueur $2r$. La distinction entre ces deux cas correspond au fait que le « milieu » d'un palindrome de longueur impaire est le caractère central, tandis que le « milieu » d'un palindrome le longueur paire est entre les deux caractères centraux.
4. Variant : $2n - a - b$. À chaque tour, soit a soit b est incrémenté, avec en outre $a \leq b \leq n$. On a précisément $2n - 1$ tours pour une chaîne de longueur > 0.
5. Variant : $a - r$. On a au minimum une comparaison de caractères (au premier tour de boucle on a $r = 0$ et les conditions $a - r \geq 0$ et $b + r < n$ sont donc systématiquement vérifiées). Ce cas est réalisé lorsque $b = a + 1$ et $s[a] \neq s[b]$. Le nombre de comparaisons est borné par la plus courte des longueurs des deux intervalles $[0, a]$ et $[b, n[$, d'où $\min(a + 1, n - b)$. Ce cas est réalisé à chaque fois qu'on explore un palindrome allant jusqu'à l'une des extrémités du mot.
6. Nombre de comparaisons compris entre $2n - 1$ (au moins une comparaison par tour pendant $2n - 1$ tours) et $\frac{n+1}{2} \times (2n - 1)$ (au plus $\min(a + 1, n - b)$ comparaisons par tour, avec $\min(a + 1, n - b) \leq (n + 1)/2$). Note : on peut encore affiner ces bornes, mais on gardera dans tous les cas une borne inférieure linéaire, et une borne supérieure quadratique.