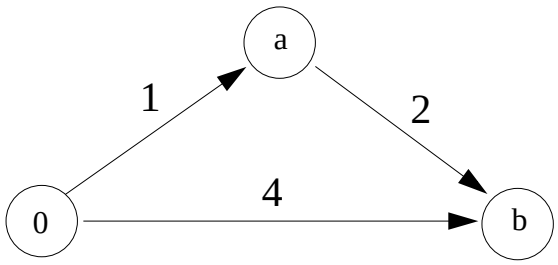
CORRECTION DE L'EXAMEN D'ALGORITHMIQUE ET COMPLEXITE

Master Informatique, première année, janvier 2015

TOUS VOS DOCUMENTS SUR PAPIER SONT AUTORISES.
COURRIEL ET TELEPHONE SONT INTERDITS.
REPONDEZ AUX QUESTIONS DANS L'ORDRE.
NUMEROTEZ VOS REPONSES.

1 Plus court chemin et programmation linéaire

Considérez le graphe ci-dessous :



Le coût (ou longueur) de l'arc $0a$ est 1, celui de l'arc ab est 2, celui de $0b$ est 4.

Exprimez le calcul des distances (longueurs des plus courts chemins) des sommets a et b au sommet source 0 comme un problème de programmation linéaire. Appelez a la distance du sommet a à la source. Appelez b la distance du sommet b à la source. Nommez a', b', b'' les variables d'écart. Résolvez par la méthode du simplexe ce problème de programmation linéaire.

Réponse.

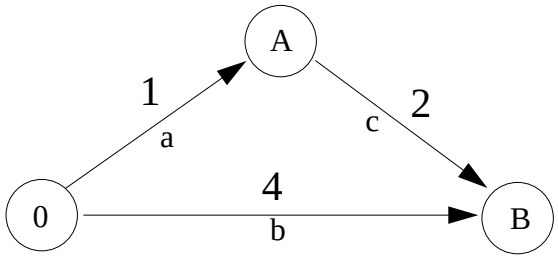
$$\begin{aligned}
&\max : a + b \\
&a \leq 1 \Rightarrow a + a' = 1 \\
&b \leq 4 \Rightarrow b + b' = 4 \\
&b \leq a + 2 \Rightarrow b - a + b'' = 2
\end{aligned}$$

Résolvons par la méthode du simplexe :

$$\begin{array}{lll}
\max : a + b & \max : 1 - a' + b & \max : 4 - 2a' - b' \\
a' = 1 - a & a = 1 - a' & a = 1 - a' \\
b' = 4 - b & b' = 4 - b & b = 3 - a' - b' \\
b'' = 2 + a - b & b'' = 3 - a' - b & b' = 1 + a' + b''
\end{array}$$

C'est terminé. On trouve bien les distances correctes : $a = 1, b = 3$.

2 Flot optimal et programmation linéaire



Pour le même graphe que précédemment, considérez le calcul de la distance de B au sommet source O comme un problème de flot de valeur 1 et de coût minimal. Posez le problème de programmation linéaire. Résolvez par la méthode du simplexe ce problème de programmation linéaire : seuls les tableaux initial et final sont demandés.

Réponse.

a est le flot dans l'arc $O \to A$; il est aussi égal au flot c dans l'arc $A \to B$. b est le flot dans l'arc $O \to B$. Le problème de PL est :

$$\begin{array}{l} \min : 3a + 4b \Rightarrow \max : -3a - 4b \\ a + b = 1 \end{array}$$

Le tableau initial et final est :

$$\begin{array}{l} \max : -3 - b \\ a = 1 - b \end{array}$$

Interprétation. Le chemin le plus court entre O et B est $O \to A \to B$. L'arc $O \to B$ est inutilisé.

Voici une seconde solution, moins expéditive. Soit c le flot dans l'arc $A \to B$. La règle de conservation du flot (en chaque sommet, la somme des flots entrants égqle la somme des flots sortants, sauf au sommet source et au sommet puits) au sommet A donne : $a = c$. Le problème de PL est :

$$\begin{array}{l} \min : a + 2c + 4b \Rightarrow \max : -a - 2c - 4b \\ a = c \\ a + b = 1 \end{array}$$

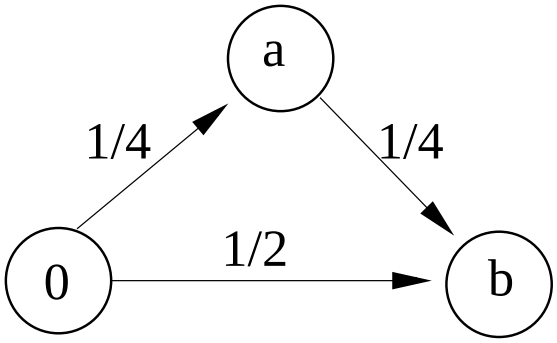
L'exemple est très simple, donc on trouve facilement un tableau initial et final :

$$\begin{array}{l} \max : -3 - b \\ c = 1 - b \\ a = 1 - b \end{array}$$

Interprétation. $a = c = 1$; $b = 0$; le coût est 3.

3 Graphes et probabilités

Par définition, la somme de la probabilité de mort et de la probabilité de survie pour un arc ou un chemin vaut 1. Dans le graphe suivant, chaque arc est étiqueté avec la probabilité de mourir quand l'arc est emprunté. Quelles sont les probabilités de survie des deux chemins $0, a, b$ et $0, b$? Quel est le chemin le plus sûr ? Aucune justification n'est demandée, et le recours à la programmation linéaire n'est pas requis.



Réponse. La probabilité de survie du chemin $0 \to a \to b$ est $(1 - 1/4) \times (1 - 1/4) = 9/16 = 0.5625$. Celle du chemin $0 \to b$ est $1 - 1/2 = 1/2$. Donc le chemin $0 \to a \to b$ a une probabilité de survie plus grande : il est plus sûr.

Commentaire : imaginez qu'il y a 16 personnes au sommet 0. Elles vont vers le sommet a : un quart meurt, donc 12 arrivent en vie en a. Un quart de ces 12 personnes meurent entre a et b, donc 9 arrivent saines et sauves en b. Le taux de survivants est donc 9/16.

4 Problème de la somme

n entiers naturels (donc non négatifs) e_i, avec $i \in 1..n$, sont donnés, ainsi qu'un entier naturel S.

a. Comment décider si S est la somme d'un des sous-ensembles des e_i ? Chaque entier e_i ne peut être utilisé qu'une seule fois. Réduisez ce problème au problème du sac à dos, que nous avons résolu par programmation dynamique.

Réponse. Pour réduire au problème du sac à dos : $\max \sum_i x_i u_i$ avec $\sum_i x_i p_i \leq S$, $x_i \in \{0, 1\}$, $i \in \{1, \ldots n\}$, prendre les utilités u_i et les poids (ou volumes) p_i égaux aux entiers e_i : $u_i = p_i = e_i, \forall i$.

Ci-dessous, une réponse plus détaillée, qui n'était pas demandée.

Dans le problème du sac à dos, un alpiniste doit choisir quels éléments prendre dans son sac à dos. Chaque élément i a une utilité $u_i \geq 0 \in \mathbb{R}$ et un poids $p_i \in \mathbb{N}$. L'alpiniste doit maximiser la somme des utilités $\sum x_i u_i$ ($x_i \in \{0, 1\}$) des éléments du sac à dos, avec la contrainte que la somme des poids $\sum x_i p_i$ des éléments du sac ne dépasse pas un poids maximal donné S. Si $e_i = u_i = p_i$, alors

ce problème est égal au problème de la somme : max : $\sum x_i e_i$ avec la contrainte $\sum x_i e_i \leq S$, $i = 1, \ldots n$, et $x_i \in \{0, 1\}$.

Le problème de la somme peut être résolu avec l'algorithme par programmation dynamique (vu en TD et programmé en TP). Notons $\Omega(i, s)$ la somme maximale plus petite ou égale à $s \in \mathbb{N}$ faisable avec les i premiers éléments (donc $i \in 1 \ldots n$). Alors

$$\begin{aligned}
&\Omega(1, s) = \text{ si } e_1 \leq s \text{ alors } e_1 \text{ sinon } 0 \\
&\Omega(i > 1, s) = \text{ si } s - e_i \geq 0 \\
&\qquad \text{alors } \max(\Omega(i-1, s), e_i + \Omega(i-1, s-e_i)) \\
&\qquad \text{sinon } \Omega(i-1, s)
\end{aligned}$$

Les valeurs de Ω peuvent être stockées dans une table de hachage. Ceci permet de ne les calculer qu'une seule fois, et seulement quand ceci est nécessaire. Ceci simplifie aussi le programmation : il suffit de demander la valeur voulue de Ω, sans que le programmeur ait se soucier de l'ordre des calculs.

b. La méthode fonctionne-t-elle pour des entiers relatifs (dans $\mathbb{Z}$) ? On suppose que S reste un entier positif.

Réponse. Non. Voici un exemple simple où la méthode précédente ne fonctionne pas avec des e_i entiers négatifs. Supposons $i = 1$, $e_1 < 0$, et $S \geq 0$, alors le résultat de la méthode est $\Omega(1, S) = e_1$, mais le résultat correct (de $\max \sum x_i e_i$ avec $\max \sum x_i e_i \leq S$) est 0.

NB. Cela ne prouve pas qu'aucune méthode de programmation dynamique n'existe pour résoudre ce problème. Cela prouve seulement que la méthode précédente ne fonctionne pas.

Le problème de la somme (ou du sac à dos) peut aussi être résolu par recherche arborescente (*backtrack*), avec élagage (*branch and bound*) avec des e_i réels et éventuellement négatifs.

5 Variante de la transformée de Fourier rapide

Arthur propose une variante de la transformée de Fourier rapide. Il considère les racines 3^k ièmes de l'unité (racines cubiques, 9 ième, 27 ième, etc) alors que l'algorithme classique considère les racines 2^k de l'unité. En conséquence, le temps d'exécution de son algorithme vérifie les relations de récurrence : $T(1) = 1, T(n = 3^k) = n + 3T(n/3)$.

a. Quelle est la complexité de la méthode d'Arthur ? La preuve par récurrence n'est pas demandée.

Réponse. Il suffisait de répondre $O(n \log n)$.

Voici une réponse détaillée, qui n'était pas demandée. On devine sur le ta-

bleau suivant que $T(n = 3^k) = (k + 1)n = O(n \log n)$:

k	$n = 3^k$	$T(n)$	$(k + 1)n$
0	1	1	1
1	3	6	6
2	9	27	27
3	27	108	108

Prouvons-le par récurrence. On sait que $T(n) = (k + 1)n$ pour $n = 3^k$ pour de petites valeurs de n. Prouvons que la propriété est vraie pour $n' = 3n = 3^{k'}$ avec $k' = k + 1$:

$$T(n') = T(3n) = 3n + 3T(n) = 3n + 3(k + 1)n = (k + 2)(3n) = (k' + 1)n'$$

CQFD.

Est-elle meilleure que celle de la méthode classique ? Donnez la complexité de la méthode classique.

Réponse. La méthode classique est elle aussi en $O(n \log n)$. La méthode classique et celle d'Arthur ont la même complexité.

6 Un graphe non orienté est-il un arbre ?

Rappel : En théorie des graphes, un arbre est un graphe non orienté, acyclique et connexe.

Soit A le nombre d'arêtes de G et S le nombre de sommets de G.

a. Quelle relation lie A et S quand G est un arbre ?

Réponse. $S = A + 1$. Remarque : l'arbre doit être non vide ($S > 0$).

b. Quelle relation lie A, S, T quand G est une forêt avec T arbres ?

Réponse. $S = A + T$. Ce coup-ci, la formule fonctionne même si $S = 0$.

c. Citez un des (au moins trois) algorithmes vus en cours permettant de décider que G est connexe. Vous pouvez supposer que la relation de (a) est vérifiée.

Réponse. 1, par gestion des classes d'équivalence (*union find*). 2 : les parcours (ou traversées) d'un graphe (en largeur ou en profondeur) permettent de construire une forêt couvrante (ici un arbre couvrant) en temps linéaire avec la taille du graphe (nombre de sommets + nombre d'arcs ou arêtes). Le livre "Introduction à l'algorithmique" de Cormen et al présentent ces deux méthodes. La première a été vue en CM. La seconde a été vue en TD.

d. On suppose que G est non orienté, acyclique et connexe (c'est un arbre). Proposez en 5 lignes au plus un algorithme calculant tous les plus courts chemins (et leurs longueurs) depuis un sommet donné. Votre algorithme doit être plus efficace que celui de Dijkstra.

Réponse. Par un parcours en profondeur à partir du sommet source ω, calculer un arbre couvrant : ceci oriente les arêtes de la source ω vers tous les autres sommets. Il suffit ensuite de propager : $D(\omega) = 0, D(s) =$ soit $p =$ father(s) dans $D(p) + L(ps)$. Le tableau D contient la distance à la source, et $L(ps)$ est la longueur de l'arc $p \to s$. Le temps est en $O(A+S) = O(A) = O(S)$, meilleur que la méthode de Dijkstra (qui est plus générale).

Un algorithme par programmation dynamique (comme celui utilisé pour les dates au plus tôt et au plus tard) fonctionne aussi, avec les mêmes performances, ssi les arêtes sont orientées de ω vers tous les autres sommets ; cette bonne orientation est indispensable : si les arêtes donnent deux arcs opposés, il y a des cycles et l'algorithme boucle.

Remarque : les algorithmes de Prim ou Kruskal calculent un arbre couvrant optimal, et donc calculent bien un arbre couvrant, mais leur complexité n'est pas linéaire.

Remarque : les algorithmes fonctionnent si des coûts sont négatifs et donnent les longueurs des chemins *simples* (sans répétition) les plus courts. Il n'y a qu'un seul chemin simple dans un arbre entre deux sommets : c'est donc le plus court. On rappelle que la longueur d'un chemin non simple peut être aussi petite que l'on veut s'il y a une arête de coût négatif dans le graphe : il suffit d'aller et venir suffisamment le long de l'arête de coût négatif.

7 Cryptographie asymétrique et bon sens

Rappel. Avec la cryptographie asymétrique, tout le monde peut crypter un message (un grand entier) m : cela revient à calculer $m' = C(m)$, ce qui se fait en temps polynomial avec $\log m$. Par contre, seul le destinataire peut décrypter m', autrement dit calculer $C^{-1}(m')$.

Un étudiant programme une de ces méthodes (par exemple RSA). Il crypte chaque bit du message, et concatène les résultats, pour obtenir le message crypté.

a. Où est l'erreur ?

Réponse. L'espion calcule $C(0)$ et $C(1)$, et regarde si le message crypté commence par $C(0)$ ou $C(1)$. Il trouve ainsi le premier bit, sans connaître la clef secrète. Il itère sur la suite du message crypté.

b. Si l'étudiant encrypte chaque octet, est-ce mieux ?

Réponse. L'espion calcule les 256 valeurs $C(0), \dots C(255)$, puis utilise la même méthode que précédemment. Ce n'est donc pas meilleur : 2^8 est trop petit.

8 Réseau

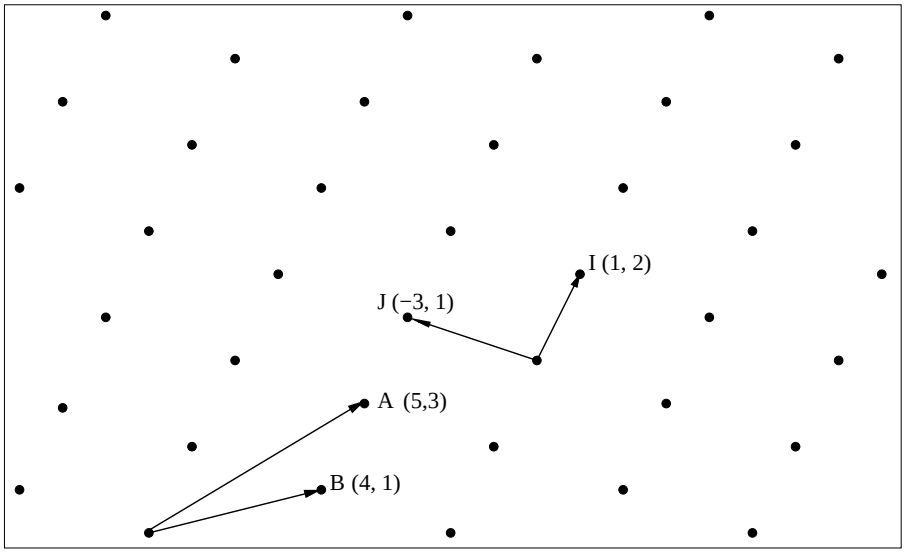
Rappels. On note $\mathbb{N}$ l'ensemble des entiers naturels : $0, 1, 2, 3 \dots$. On note $\mathbb{Z}$ l'ensemble des entiers relatifs : $\dots, -3, -2, -1, 0, 1, 2, 3 \dots$. $\mathbb{Q}$ est l'ensemble des rationnels.

Le dessin suivant montre une partie d'un réseau R généré par deux vecteurs $A = (5, 3)$ et $B = (4, 1)$. R est l'ensemble des points (les disques noirs sur le dessin) $aA + bB$ avec $a \in \mathbb{Z}$, et $b \in \mathbb{Z}$. R est aussi généré par $I = (1, 2)$ et $J = (-3, 1)$, qui sont plus courts que A et B. Il n'existe pas de base strictement plus courte que $\pm I, \pm J$.

Rappels. La longueur, ou norme euclidienne, de $A = (A_x, A_y)$ est $||A|| = \sqrt{A_x^2 + A_y^2}$. L'aire du parallélogramme généré par A, B est la valeur abosolue du déterminant

$$\begin{vmatrix} A_x & A_y \\ B_x & B_y \end{vmatrix} = A_x B_y - A_y B_x$$

Propriété : toutes les bases d'un même réseau génèrent des parallélogrammes de même aire (au signe près).



a. Deux vecteurs 2d indépendants à coordonnées dans $\mathbb{Z}$: $A = (A_x, A_y)$ et $B = (B_x, B_y)$, sont donnés. Proposez un test (utilisant un nombre constant d'opérations dans $\mathbb{Z}$) pour décider qu'il n'existe pas de vecteurs strictement plus courts que $\pm A, \pm B$ engendrant le même réseau. Vous pouvez supposer $||A|| \geq ||B||$. Répondre en une ligne au plus.

Réponse. Supposons $||A|| \geq ||B||$. Si ni $A + B$ ni $A - B$ n'est strictement plus court que A, alors $(\pm A, \pm B)$ est la base la plus courte.

b. Déduisez-en un algorithme de calcul de la base la plus courte. Quel est son coût ?

Réponse. On peut échanger A et B pour que A ne soit pas plus court que B. Soit C le vecteur le plus court dans $\{A+B, A-B\}$. Si C est plus court que A, itérer sur la base (C, B) sinon (A, B) est la base la plus courte. Cette méthode n'est pas en temps polynomial : considérer $B = (1, 0)$ et $A = (\alpha \in \mathbb{N}, 1)$.

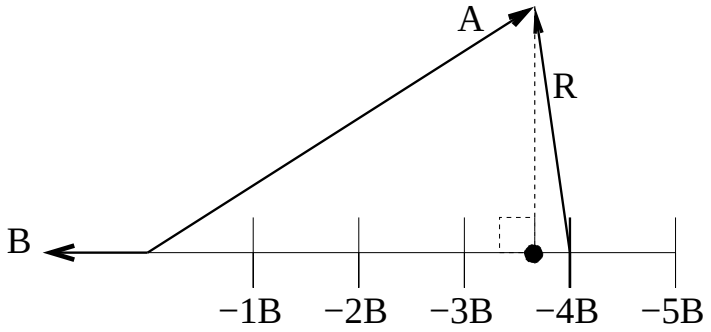
Alors la base la plus courte est $I = (1,0), J = (0,1)$ mais $O(\alpha)$ réductions sont nécessaires, ce qui est exponentiel dans la taille des données (la taille des données est $\log \alpha$).

La suite n'était pas demandée.

Voici une méthode plus efficace.

Soit $P = qB$ la projection orthogonale de A sur B (avec $q \in \mathbb{R}$; alors $A - qB$ et B sont perpendiculaires :

$$(A - qB) \,.\, B = 0 \Rightarrow q = \frac{A \,.\, B}{B \,.\, B}$$



Rappel : $A \,.\, B$ est le produit scalaire de A et B. Il vaut $A_x B_x + A_y B_y$.

Soit $q_e = \lfloor q \rfloor$ ou bien $q_e = \lfloor q \rceil$ l'entier le plus proche de q (arrondir arbitrairement si $q \in \mathbb{Z}/2$). Alors soit le "reste" $R = A - q_e B$; dans la base (A, B), A est remplacé par le vecteur le plus court dans $\{A, R\}$; la méthode termine quand R n'est pas strictement plus court que A.

Chaque étape de réduction de la base s'exprime matriciellement :

$$\begin{pmatrix} B_x & B_y \\ R_x & R_y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_e \end{pmatrix} \begin{pmatrix} A_x & A_y \\ B_x & B_y \end{pmatrix} \text{ avec } q_e = \left\lfloor \frac{A \,.\, B}{B \,.\, B} \right\rceil \text{ ou } \left\lfloor \frac{A \,.\, B}{B \,.\, B} \right\rfloor$$

ou bien, si on préfère :

$$\begin{pmatrix} q_e & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} B_x & B_y \\ R_x & R_y \end{pmatrix} = \begin{pmatrix} A_x & A_y \\ B_x & B_y \end{pmatrix} \text{ avec } q_e = \left\lfloor \frac{A \,.\, B}{B \,.\, B} \right\rceil \text{ ou } \left\lfloor \frac{A \,.\, B}{B \,.\, B} \right\rfloor$$

et la matrice contenant q_e a comme déterminant -1. Elle est unimodulaire, ce qui prouve que toutes les bases ont même déterminant (au signe près) que la base initiale (A, B). Cette formulation matricielle fonctionne aussi quand B est plus court que A, et dans ce cas, q_e est nul et la réduction échange A et B ; ce cas ne peut se produire deux fois de suite, et il n'influence donc pas l'ordre de grandeur du nombre de réductions.

Soulignons l'analogie avec l'algorithme d'Euclide étendu, qui calcule le PGCD g de deux nombres a et b, et les coefficients de Bézout x et y tels que $ax + by = g$. Cet algorithme pose : $a = qb + r$ (nous utilisons des minuscules pour éviter les

confusions) avec $q = \lfloor a/b \rfloor$ (utiliser $q = \lfloor a/b \rceil$ comme Gauss le faisait est correct aussi) et se formule matriciellement de la même façon :

$$\begin{pmatrix} b \\ r \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \text{ avec } q = \left\lfloor \frac{a}{b} \right\rceil \text{ ou } \left\lfloor \frac{a}{b} \right\rfloor$$

ou bien, si on préfère :

$$\begin{pmatrix} q & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} b \\ r \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} \text{ avec } q = \left\lfloor \frac{a}{b} \right\rceil \text{ ou } \left\lfloor \frac{a}{b} \right\rfloor$$

Pour que tous les q_e soient positifs, comme les $q = \lfloor a/b \rfloor$ de l'algorithme classique d'Euclide, il suffit d'intercaler des phases qui remplacent le vecteur B par $-B$ quand $A \,.\, B$ est négatif ; ce changement de signe ne peut pas se produire deux fois de suite ; il ne modifie donc pas la complexité (le nombre d'étapes) et accentue la similitude entre les deux méthodes.

Cette similitude entre la réduction de la base d'un réseau 2D et l'algorithme d'Euclide nous permet de réutiliser l'étude de la complexité de l'algorithme d'Euclide (théorème de Lamé) :

Dans le cas qui nécessite le plus de réductions, que ce soit pour la méthode d'Euclide ou pour la réduction de base, tous les q, ou q_e, valent 1. Pour la méthode d'Euclide, cela se produit quand a et b sont deux nombres de Fibonacci consécutifs, comme l'a remarqué Lamé. D'ailleurs, on reconnaît alors dans la matrice contenant q_e ou q la "matrice de Fibonacci" (ou son inverse) : celle dont la puissance permet de calculer les termes de la suite de Fibonacci. La matrice de Fibonacci est F :

$$F = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = R^{-1} \begin{pmatrix} \phi & 0 \\ 0 & \phi' \end{pmatrix} R$$

avec

$$R = \begin{pmatrix} \phi & \phi' \\ 1 & 1 \end{pmatrix} \text{ et } R^{-1} = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 & -\phi' \\ -1 & \phi \end{pmatrix}$$

La valeur propre la plus grande de la matrice de Fibonacci F est $\phi = (1 + \sqrt{5})/2 \approx 1.618$. ϕ est racine de

$$det(F - \lambda I_2) = \lambda^2 - \lambda - 1 = 0, \text{ où } I_2 \text{ est la matrice identité 2 par 2}$$

La valeur propre conjuguée est $\phi' = 1 - \phi = (1 - \sqrt{5})/2 \approx -0.618$.

Par l'argument habituel, on en déduit que le nombre de réductions, ou itérations, pour les deux algorithmes est en $O(\log_\phi(N)$, où N est la norme des vecteurs initiaux de base A et B, ou bien la valeur absolue de a et b ; le nombre de réductions ou d'itérations est donc proportionnel à la taille du problème, dans le pire des cas. Le théorème de Lamé précise la constante du $O(\log N)$. Le plus souvent, le nombre d'itérations est moindre ; ce nombre d'itérations est une question importante en théorie des nombres.

Attention : chaque réduction effectue une quantité constante d'opérations dans $\mathbb{Z}$ ou $\mathbb{Q}$, mais les opérations dans $\mathbb{Z}$ ou $\mathbb{Q}$ (avec la librairie nums.cma de ocaml par exemple) ne se font pas en temps constant.

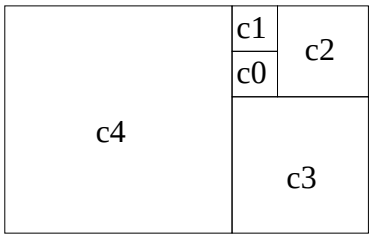
c. Le caractère entier relatif des coordonnées de A et B est-il indispensable ?

Réponse. Non. Ce qui importe est que ces coordonnées sont calculables (par exemple elles sont dans $\mathbb{Q}$, ou algébriques), que le réseau est l'ensemble des $aA + bB$ avec $a \in \mathbb{Z}, b \in \mathbb{Z}$. Cela implique que, dans un disque de rayon fini (par exemple $\max(||A||, ||B||)$) ou un carré, le réseau ne peut avoir qu'un nombre fini de points [1]. En passant, un théorème de Minkowski sur les réseaux quantifie cela. Il n'y a donc qu'un nombre fini de bases à envisager.

En haute dimension, le calcul du vecteur le plus court d'un réseau, le calcul de la base la plus courte (ou raisonnablement courte), le calcul du nombre de points d'un réseau dans un convexe sont des problèmes difficiles. Ces problèmes se posent en cryptographie, en cryptanalyse, en combinatoire, en optimisation discrète, en théorie des nombres.

9 Rectangle pavé par des carrés

Ce rectangle est pavé par des carrés. Exprimez c_1, c_2, c_3, c_4 en fonction de c_0. Que constatez vous ? Vous pouvez poser $c_0 = 1$.

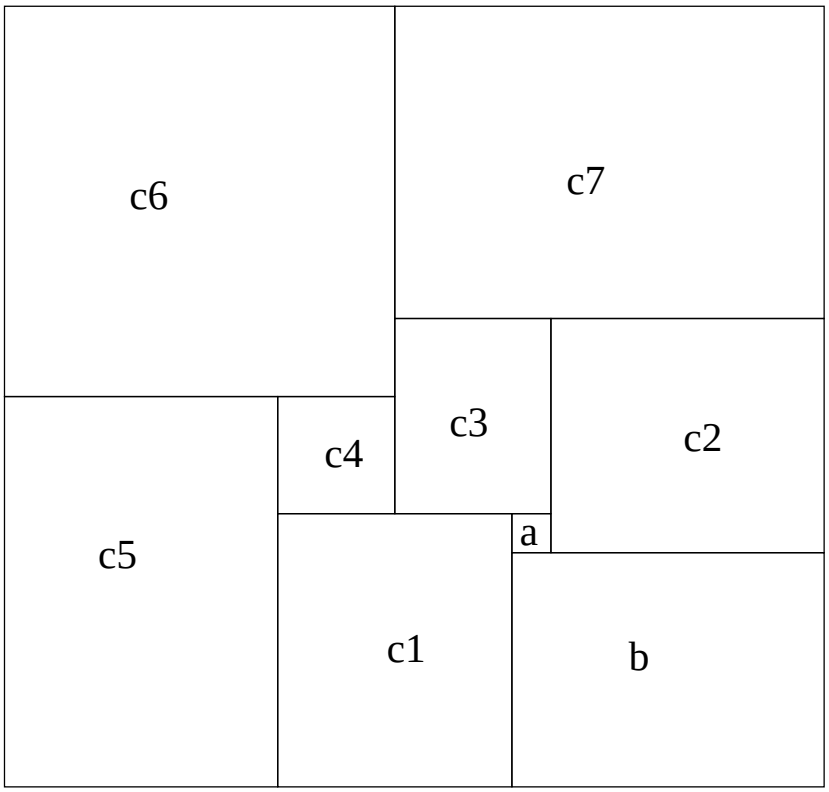


Réponse. $c_1 = c_0$. Puis $c_2 = c_0 + c_1 = 2c_0$. Puis $c_3 = c_1 + c_2 = 3c_0$. Puis $c4 = c_2 + c3 = 5c_0$. On devine que $c_k = c_{k-2} + c_{k-1}$. On reconnaît la suite de Fibonacci.

10 Rectangle pavé par des carrés tous différents

Le dessin suivant montre un rectangle pavé par des carrés ; il est faux, mais la topologie est correcte.

1. Note. Il ne suffit pas que le réseau soit discret (discontinu) pour cela. Par exemple, l'ensemble discret $1/i$ pour $i \in \mathbb{N} - \{0\}$ contient une quantité infinie de points dans l'intervalle $[0, 1]$ de longueur 1.

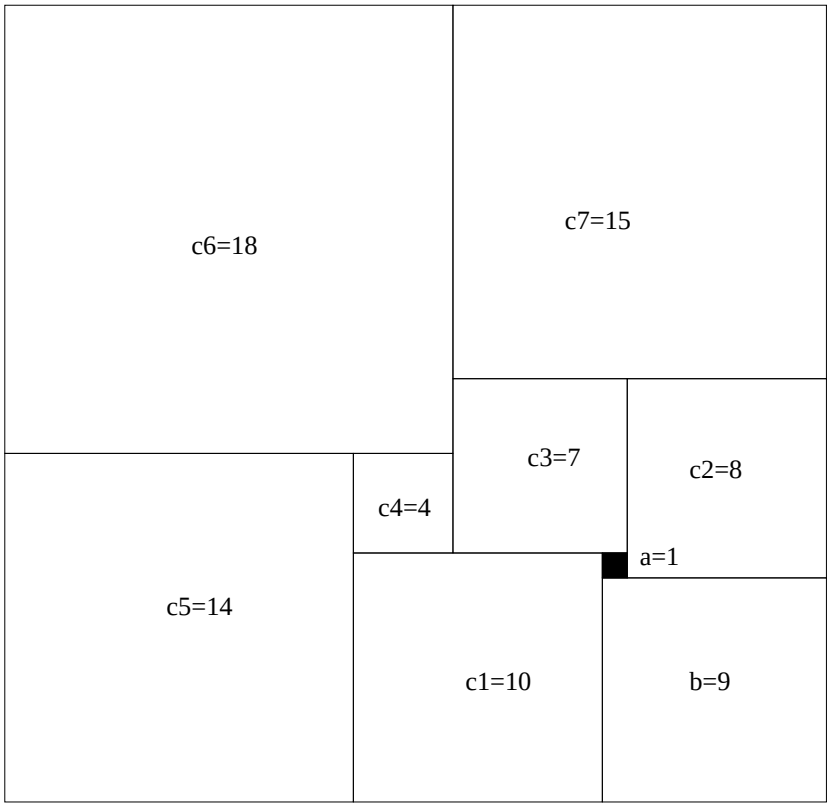


a. Exprimez les longueurs des côtés en fonction de a et b. Pour commencer : $c_1 = a+b$ et $a+c_2 = b \Rightarrow c_2 = b-a$. Pour vous aider, les carrés sont numérotés dans le "bon" ordre. Déduisez-en une relation entre a et b. Déduisez-en les longueurs entières les plus petites (non nulles) de tous les carrés.

Réponse.

$$
\begin{aligned}
c_1 &= a+b \\
c_2 &= b-a \\
c_3 &= c_2 - a = b-2a \\
c_4 &= c_1 - c_3 + a = 4a \\
c_5 &= c_1 + c_4 = 5a+b \\
c_6 &= c_4 + c_5 = 9a+b \\
c_7 &= -c_3 + c_4 + c_6 = 15a \\
c_7 &= c_2 + c_3 = 2b-3a
\end{aligned}
$$

Donc $15a = 2b-3a \Rightarrow 9a = b$. Posons $a = 1$. Alors $b = 9, c_1 = 10, c_2 = 8, c_3 = 7, c_4 = 4, c_5 = 14, c_6 = 18, c_7 = 15$.



Remarquez que, de façon plus générale, chaque trait intérieur maximal donne une équation linéaire, par exemple $c_4 + c_6 = c_3 + c_7$ pour le trait vertical en haut presque au milieu. On admettra qu'il y a toujours une équation de moins que d'inconnues.

b. Le dessin d'un pavage d'un rectangle en carrés est donné ; il faut calculer les longueurs entières minimales (non nulles) des carrés. Proposez en trois lignes au plus le principe d'une méthode, ou bien identifiez le problème mathématique sous-jacent.

Réponse. Une des longueurs est initialisée à 1. Nous admettons que le système linéaire a alors autant d'équations que d'inconnues. Il est résolu dans les rationnels ($\mathbb{Q}$). Soit C le vecteur rationnel solution ; soit p le PPCM (plus petit commun multiple) des dénominateurs des C_i ; alors pC est, algébriquement, la solution entière non nulle la plus petite du système linéaire ; mais il faut vérifier que pC est géométriquement réalisable, autrement dit que toutes les longueurs $pC_i \in \mathbb{Z}$ sont positives.

Ceci suggère une méthode pour trouver les pavages de rectangles par des carrés : générer toutes les topologies possibles, puis résoudre. C'est un sujet possible pour les projets.