SUJETS de TD d'ALGORITHMIQUE

Variante du PGCD et de l'algorithme d'Euclide étendu, calculant les nombres de Bézout

Soient a et b deux entiers naturels. Voici une méthode pour calculer leur pgcd sans utiliser de division (à part la division par 2 pour les nombres pairs).

pgcd(0, b)=b, pgcd(a, 0) = a
pgcd(2a, 2b) = 2 * pgcd(a, b)
pgcd(2a, 2b+1) = pgcd(a, 2b+1)
pgcd(a, b) = pgcd(minimum(a, b), |a-b|/2) quand a et b sont impairs (donc a-b est pair).

Faites le sur quelques exemples. Remarquez que le nombre de chiffres d'au moins un des arguments est diminué de 1 à chaque itération. En déduire la complexité dans le pire des cas.

Modifiez cette méthode pour calculer aussi les nombres (des entiers relatifs) de Bézout u et v tels que a*u+b*v=pgcd(a,b).

SOLUTION
Variante du PGCD et de l'algorithme d'Euclide étendu, calculant les nombres de Bézout

Soient a et b deux entiers naturels. Voici une méthode pour calculer leur pgcd sans utiliser de division (à part la division par 2 pour les nombres pairs).

pgcd(0, b)=b, pgcd(a, 0) = a
pgcd(2a, 2b) = 2 * pgcd(a, b)
pgcd(2a, 2b+1) = pgcd(a, 2b+1)
pgcd(a, b) = pgcd(minimum(a, b), |a-b|/2) quand a et b sont impairs (donc a-b est pair).

Faites le sur quelques exemples. Remarquez que le nombre de chiffres d'au moins un des arguments est diminué de 1 à chaque itération. En déduire la complexité dans le pire des cas.

Réponse : Le nombre d'itérations est au plus la somme du nombre de chiffres (en base 2) de a, et du nombre de chiffres de b. Le log en base 2 de a est, à 1 près, le nombre de chiffres en base 2 de a. Idem pour b. La complexité est donc en O(log(a)+log(b)). Asymptotiquement, il a la même complexité que celle d'Euclide (utilisant la division). En pratique, il peut aller un peu plus vite ; surtout il ne nécessite pas de division, ce qui peut simplifier dans certains cas (implantation matérielle, librairie sur de grands entiers).

Réponse : Modifiez cette méthode pour calculer aussi les nombres (des entiers relatifs) de Bézout u et v tels que a*u+b*v=pgcd(a,b).

Notons (u, v, g) le triplet tels que au+bv=g=pgcd(a, b).
bezout(0, b) = (0, 1, b)
bezout(2a, 2b) = (u', v', 2g') où (u', v', g')=bezout(a, b)
bezout(2a, B=2b+1)= soit (u', v', g) =bezout(a, B), donc au'+Bv'=g
    si u' est pair, alors (2a(u'/2)+Bv'=g donc u=u'/2, v=v'
    si u' est impair, alors a(u'+B)+B(v'-a)=g et u'+B est impair,
    donc (2a)((u'+B)/2)+B(v'-a)=g et u=(u'+B)/2, v=v'-a convient.
    La solution symétrique u=(u'-B)/2, v=v'+a convient aussi. On peut prendre la plus petite.
bezout(a, b)= (u'-v', v', g') où (u', v', g')=bezout(a, b-a) quand a et b sont tous les deux impairs.

BEZOUT, ET PROJECTION D'UN POINT SUR UNE DROITE

Soient a et b deux entiers naturels. Vous avez calculé g, le pgcd de a et b, ainsi que deux nombres de Bézout u et v, deux entiers relatifs tels que au+bv=g.

Proposer quelques méthodes pour calculer (U, V) tels que aU+bV=g, et (U, V) sont les plus petits possibles.

Indication. Si (u, v) est solution, alors (u+tb, v-ta) aussi, pour tout entier relatif t.

Exemple : dessiner le cas a=3, b=7. Des (u,v) solutions sont (-2,1), (5, -6), etc

Indication. ax +by=g est l'équation d'une droite qui passe par les points de Bézout (u, v). Le problème est de trouver le point entier sur cette droite qui est le plus près de l'origine (0, 0). Calculer le point qui est la projection orthogonale de (0,0) sur cette droite. Vérifier que c'est (x0,y0)=(ga/N, gb/N) où $N=a^2+b^2$.

SOLUTION
BEZOUT, ET PROJECTION D'UN POINT SUR UNE DROITE

Soient a et b deux entiers naturels. Vous avez calculé g, le pgcd de a et b, ainsi que deux nombres de Bézout u et v, deux entiers relatifs tels que au+bv=g.

Proposer quelques méthodes pour calculer (U, V) tels que aU+bV=g, et (U, V) sont les plus petits possibles.

Indication. Si (u, v) est solution, alors (u+tb, v-ta) aussi, pour tout entier relatif t.

Exemple : dessiner le cas a=3, b=7. Des (u,v) solutions sont (-2,1), (5, -6), etc

Indication. ax +by=g est l'équation d'une droite qui passe par les points de Bézout (u, v). Le problème est de trouver le point entier sur cette droite qui est le plus près de l'origine (0, 0). Calculer le point qui est la projection orthogonale de (0,0) sur cette droite. Vérifier que c'est (x0,y0)=(ga/N, gb/N) où $N=a^2+b^2$.

Solution possible. La droite a comme équation paramétrique : x=u+tb, y=v-ta. Calculer pour quelle valeur de t le point x,y est la projection de l'origine sur la droite.

Première solution : u+tb=ga/N, donc t= (ga/N-u)/b, où $N=a^2+b^2$.
Une autre solution, équivalente : v-ta=gb/N donc t=(v-gb/N)/a . L'étudiant vérifiera l'égalité.

Ce t n'est pas entier. Il est intuitivement clair que le point solution est obtenu en considérant soit la partie entière de t, soit 1 + la partie entière.

Une autre solution. Trouver t tel que $f(t)=(u+tb)^2+(v-ta)^2$ est minimal. Considérer cette expression comme une fonction de t, disons f(t). Elle est minimale (ou maximale, mais nous vérifierons après) quand la dérivée f' s'annule. $f'(t)=2(u+tb)b-2a(v-ta) = 2(ub-av + t(a^2+b^2)$. Finalement, f'(t)=0 implique $t=(av-bu)/(a^2+b^2)$. Choisir entre l'arrondi entier de t par défaut, et l'arrondi entier de t par excès, qui donne le point entier (U,V) le plus proche de l'origine.

VARIANTE DU TRI : LE FRONT DE PARETO

Un ensemble fini de points en 2D est donné. Les coordonnées sont des entiers naturels (donc non négatives). On dit qu'un point A est meilleur qu'un point B ssi l'abcisse de A est plus petite ou égale à celle de B, et l'ordonnée de A est plus petite ou égale à celle de B. Le problème est de calculer l'ensemble des meilleurs points (il n'en existe pas de meilleurs), et de les trier par x croissant. Cet ensemble s'appelle un front de Pareto ; il survient en optimisation multicritère, ici la minimisation de x et celle de y. Un point A appartient au front de Pareto, quand il n'existe pas de point ayant en même temps une abscisse et une ordonnée plus petites que celles de A.

Dessiner quelques exemples. S'il y a n points, combien de points peut avoir le FP ?

Proposez une méthode pour générer des points au hasard au dessus de la diagonale descendante du carré [0, 512] par [0, 512]. Il existe vraisemblablement une fonction random(M) qui rend un entier pseudo-aléatoire entre 0 et M-1.

Généralisez d'abord le tri rapide. Pour cela, proposez une méthode non déterministe (recourant au hasard) pour trouver un pivot, c'est à dire un point du FP en O(n). Ensuite, éliminez les points d'abscisse et d'ordonnée plus grandes que celles du pivot. Faire deux appels récursifs, pour calculer le FP de l'ensemble des points à gauche et au dessus du pivot, et le FP de l'ensemble des points à droite et en dessous du pivot. Concaténez les résultats.

Généralisez le tri fusion. Pour cela, il vous faut généraliser la fusion de deux FP (qui sont ordonnés par x croissant).

SOLUTION
Variante du tri : le front de Pareto.

Un ensemble fini de points en 2D est donné. Les coordonnées sont des entiers naturels (donc non négatives). On dit qu'un point A est meilleur qu'un point B ssi l'abcisse de A est plus petite ou égale à celle de B, et l'ordonnée de A est plus petite ou égale à celle de B. Le problème est de calculer l'ensemble des meilleurs points (il n'en existe pas de meilleurs), et de les trier par x croissant. Cet ensemble s'appelle un front de Pareto ; il survient en optimisation multicritère, ici la minimisation de x et celle de y. Un point A appartient au front de Pareto, quand il n'existe pas de point ayant en même temps une abscisse et une ordonnée plus petites que celles de A.

Dessiner quelques exemples. S'il y a n points, combien de points peut avoir le FP ?

Réponse. Il y en a entre 1 et n. Il ne peut pas y en avoir 0.

Proposez une méthode pour générer des points au hasard au dessus de la diagonale descendante du carré [0, 512] par [0, 512]. Il existe vraisemblablement une fonction random(M) qui rend un entier pseudo-aléatoire entre 0 et M-1.

Réponse. Triviale.

Généralisez d'abord le tri rapide. Pour cela, proposez une méthode non déterministe (recourant au hasard) pour trouver un pivot, c'est à dire un point du FP en O(n).

Réponse. Parcourir la liste des points P1, P2... Pn en initialisant le meilleur point P à P1. Comparer le meilleur point P à chaque Pi. Si l'un des deux est meilleur que l'autre, mettre à jour P:= meilleur(P, Pi). Si aucun des 2 n'est meilleur que l'autre, tirer à pile ou face la nouvelle valeur de P : c'est l'ancienne valeur avec probabilité ½, et Pi avec probabilité ½. A la fin, P contient un élément du FP.

Ensuite, éliminez les points d'abscisse et d'ordonnée plus grandes que celles du pivot. Faire deux appels récursifs, pour calculer le FP de l'ensemble des points à gauche et au dessus du pivot, et le FP de l'ensemble des points à droite et en dessous du pivot. Concaténez les résultats.

Généralisez le tri fusion. Pour cela, il vous faut généraliser la fusion de deux FP (qui sont ordonnés par x croissant).

Réponse. La fusion de 2 fronts ordonnés par x croissant est proche de la fusion de deux listes triées. Représentons les deux fronts par des listes. Dans A=A1::Aq, A1 est la tête de liste, Aq la queue.
Fusion(vide, B) = B, fusion(A, vide)=A
Fusion(A= A1::Aq, B= B1::Bq) =

```
    si A1 est meilleur que B1 alors A1::(Fusion Aq Bq)   --il faut éliminer B1
    si B1 est meilleur que A1 alors B1::(Fusion Aq Bq)   --il faut éliminer A1
    si aucun n'est meilleur que l'autre                  -- ils appartiennent tous les 2 au FP fusion
        si A1.x < B1.x alors A1::(Fusion Aq B)           –le FP est trié par x croissant
        sinon B1::(Fusion A Bq)                          –le FP est trié par x croissant
```

La complexité de la fusion est proportionnelle au nombre de points. La partition en deux moitiés est réalisée comme pour le tri par fusion. Cette méthode est toujours en O(n log n), même si il n'y a qu'un seul point dans le FP final. Peut-être existe-t-il une méthode en O(n + f log f), où f est le nombre de points dans le front de Pareto.

PARETO, suite.

Dans le problème du front de Pareto, il peut arriver que l'ensemble des points soit infini. Cependant, même quand l'ensemble des points (à coordonnées entières, non négatives) est infini, le front de Pareto a un nombre fini de points.

Prouver le.

Pourquoi est-ce important ?

Ce théorème, généralisé en dimension quelconque, est le lemme de Dickson : l'ensemble des éléments minimaux de tout ensemble (même infini) de vecteurs d'entiers naturels est fini. Ce lemme permet de prouver la terminaison de l'algorithme des bases de Grobner (1965, dû à Bruno Buchberger), utilisé en calcul formel (Mathematica, Mapple, etc).

PARETO, suite.

Dans le problème du front de Pareto, il peut arriver que l'ensemble des points soit infini. Cependant, même quand l'ensemble des points (à coordonnées entières, non négatives) est infini, le front de Pareto a un nombre fini de points.

Prouver le.

Pourquoi est-ce important ?

Ce théorème, généralisé en dimension quelconque, est le lemme de Dickson : l'ensemble des éléments minimaux de tout ensemble (même infini) de vecteurs d'entiers naturels est fini. Ce lemme permet de prouver la terminaison de l'algorithme des bases de Grobner (1965, dû à Bruno Buchberger), utilisé en calcul formel (Mathematica, Mapple, etc).

Preuve (pour le cas 2D)

Soit un point P du front.

A gauche de P, il n'y a qu'une quantité finie de droites verticales, à abscisses entières, disons G : au plus, il peut y avoir seulement G sommets du front de Pareto à gauche de P ; en effet, il y a au plus un point de Pareto sur une droite verticale d'abscisse entière.

En dessous de P, même raisonnement : il n'y a qu'une quantité finie de droites horizontales d'ordonnée entière.