MARO 037 - Competitive Programming

Fabian Lecron - Arnaud Vandaele

Faculté Polytechnique - Services MIT et MARO

UMONS

Année académique 2024-2025

Séance 05 :
Quelques thèmes en théorie des nombres

Séance 05 – Quelques thèmes en théorie des nombres

Tables des matières

Plan

Compter le nombre de chiffres à l'aide de `log`

Formule principale : le nombre de chiffre d'un nombre x en base b est

$$\lfloor \log_b(x) \rfloor + 1$$

Questions :

- Trouver l'entier n pour que le nombre 17^n s'écrive avec 20 chiffres ?
- Pour les nombres n suivants, combien de chiffres sont nécessaires pour écrire $n!$ en base 10

Input	Output
0	
1	
3	
10	
20	
1000000	

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \quad \text{(Stirling's approximation)}$$

Plan

Nombres premiers : vocabulaire et définitions

- La notation $d|a$ (d *divise* a) signifie que $a = kd$ pour un entier k.
- Un **diviseur** d'un entier $a \neq 0$ vaut au moins 1 mais n'est pas supérieur à $|a|$ (par exemple, les diviseurs de 24 sont $1, 2, 3, 4, 6, 8, 12, 24$)
- Tout entier a est divisible par les **diviseurs triviaux** 1 et a.

  Les diviseurs non triviaux de a sont appelés **facteurs** de a.

  (les facteurs de 24 sont $2, 3, 4, 6, 12$ et ses diviseurs triviaux sont 1 et 24)
- Un entier $a > 1$ est un **nombre premier** si ses seuls diviseurs sont les diviseurs triviaux.

Pour tout entier $n > 1$, il existe une factorisation unique en nombres premiers :

$$n = p_1^{\alpha_1} p_2^{\alpha_2} ... p_k^{\alpha_k},$$

où les p_i sont des nombres premiers distincts et les α_i des entiers positifs.
Exemple pour $n = 84$:

$$84 = 2^2.3^1.7^1.$$

Questions :

- Comment vérifier qu'un nombre est premier ?
- Comment obtenir les facteurs de la factorisation en nombres premiers ?

Vérifier qu'un nombre n est premier

- Idée de base :

  une solution simple consiste à parcourir tous les nombres x de 2 à $n-1$ et pour chaque nombre, on vérifie s'il divise n.

  Si on trouve un nombre x qui divise n, on renvoie `False`.

  (complexité : $\mathcal{O}(n)$)

- Idée plus efficace :

  au lieu de vérifier jusqu'à $n-1$, on peut ne vérifier que jusqu'à $\lfloor\sqrt{n}\rfloor$ car un facteur plus grand que $\lfloor\sqrt{n}\rfloor$ doit alors être multiplié par un facteur plus petit qui a forcément déjà été vérifié. (complexité : $\mathcal{O}(\sqrt{n})$)

- Idée encore un peu améliorée :

  L'algorithme peut être encore amélioré en observant que tous les nombres premiers sont de la forme $6k \pm 1$, à l'exception de 2 et 3. En effet, tous les entiers peuvent être exprimés comme $(6k+i)$ pour un entier k et pour $i = -1, 0, 1, 2, 3$ ou 4. De plus, 2 divise $(6k+0)$, $(6k+2)$, $(6k+4)$ et 3 divise $(6k+3)$. Une méthode plus efficace consiste donc à tester si n est divisible par 2 ou 3, puis à vérifier tous les nombres de la forme $6k \pm 1$.

- Et d'autres, encore (bien) plus efficaces

Calculer les facteurs premiers d'un entier

La méthode la plus simple est appelée "*Trial Division*".
Cette méthode consiste à systématiquement tester tous les facteurs possibles et à vérifier s'ils divisent réellement le nombre donné.

- Il vaut mieux tester les diviseurs par ordre croissant car un nombre n arbitraire est plus susceptible d'être divisible par 2 que par 3, etc.
- Avec cet ordre, il ne sert à rien de tester la divisibilité par 4 si le nombre a déjà été déterminé non divisible par 2, et ainsi de suite pour 3 et tout multiple de 3, etc. Par conséquent, l'effort peut être réduit en sélectionnant uniquement les nombres premiers comme facteurs candidats.
- De plus, comme au slide précédent, il n'est pas nécessaire d'aller plus loin que $\sqrt{n}$

Crible d'Ératosthène (sieve of Eratosthenes)

Le crible d'Ératosthène est un algorithme permettant de trouver tous les nombres premiers inférieurs à un certain n.
Il fonctionne de la manière suivante :

1. On crée un tableau `sieve` qui permettra de vérifier facilement si un entier entre 2 et n est premier ou non (si x est premier, alors on aura `sieve[x]=1`, sinon `sieve[x]=0`)
2. Initialement, on fixe $x = 2$, le plus petit nombre premier
3. On énumère ensuite tous les multiples de x ($2x$, $3x$, $4x$, ...) et on indique qu'ils ne sont pas premiers
4. On cherche ensuite le premier nombre supérieur à x dans la liste qui n'est pas encore marqué. S'il n'y en a pas, on arrête. Sinon, on fixe x à cette nouvelle valeur et on retourne à l'étape 3.
5. Lorsque l'algorithme se termine, les nombres restants non marqués dans la liste sont tous les nombres premiers inférieurs à n.

Complexité : il est possible de montrer que l'algorithme fonctionne en $\mathcal{O}(n \log \log n)$

Plan

Calcul du plus grand commun diviseur

Le plus grand commun diviseur des entiers *a* et *b*, noté `gcd(a,b)` est le plus grand entier qui divise à la fois *a* et *b*. Par exemple, $gcd(30, 12) = 6$.

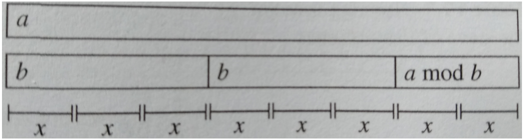
- Une manière de calculer `gcd(a,b)` consiste à calculer la décomposition en facteurs premiers de *a* et de *b* et d'ensuite choisir l'exposant le plus grand apparaissant dans les deux factorisations pour chaque facteur.

  Exemple : $30 = 2.3.5$ et $12 = 2^2.3$, donc

  $$gcd(30, 12) = 2.3 = 6$$

- L'algorithme d'Euclide est beaucoup plus efficace et se base sur le constat suivant : *si on soustrait du plus grand nombre le plus petit nombre, le plus grand commun diviseur ne change pas*. Exemple: $gcd(30, 12) = gcd(18, 12)$.

  En version récursive, nous avons donc : $gcd(a, b) = gcd(b, a \bmod b)$.



Algorithme d'Euclide étendu

Question : quels sont les entiers x et y qui permettent de satisfaire l'équation

$$30x + 12y = 6$$

De façon générale, quels entiers x et y permettent de satisfaire l'équation

$$ax + by = gcd(a, b)$$

Ce problème est résolu par récurrence.
Supposons qu'on connaisse la solution (x', y') du problème

$$bx' + (a \bmod b)y' = gcd(a, b)$$

Puisque $a \bmod b = a - \lfloor \frac{a}{b} \rfloor b$, l'équation ci-dessus se transforme alors en

$$ay' + b(x' - \lfloor \frac{a}{b} \rfloor y') = gcd(a, b)$$

ce qui fournit donc la solution au problème de départ : $x = y'$ et $y = x' - \lfloor \frac{a}{b} \rfloor y'$.

Equation diophantienne linéaire

Résolution à l'aide de l'algorithme d'Euclide étendu

Une équation diophantienne linéaire est de la forme

$$ax + by = c$$

où a, b et c sont des entiers donnés et on recherche, si possible, des entiers x et y afin de satisfaire l'équation.
Exemple : est-il possible de trouver une solution à l'équation

$$39x + 15y = 12$$

Exercice. Clara veut acheter des pizzas et des sodas. Elle a 400 euros. Si nous savons que chaque pizza coûte 57 euros et chaque bouteille de soda coûte 22 euros, combien de pizzas et de bouteilles de soda peut-elle acheter?

$$57x + 22y = 400$$

Plan

Exemple Kattis 1 : Fruit Baskets - Description

Problème : https://open.kattis.com/problems/fruitbaskets, de difficulté 4.6.

Données : un entier N et une série d'entiers n_i, avec $i = 1, ..., N$.

exemple : $N = 4$ et $n = [50 \quad 60 \quad 70 \quad 120]$

Description du problème :

- Chaque entier n_i représente le poids d'un type de fruit
- En effectuant les différentes combinaisons possibles de ces fruits, il y a moyen de construire $2^N - 1$ paniers différents (il y a -1 car on ne considère pas le cas du panier vide). Pour l'exemple, cela donne 15 paniers possibles :

$\{50\}, \{60\}, \{70\}, \{120\}, \{50, 60\}, \{50, 70\}, \{50, 120\}, \{60, 70\}, \{60, 120\}, \{70, 120\}, \{50, 60, 70\}, \{50, 60, 120\}, \{50, 70, 120\}, \{60, 70, 120\}, \{50, 60, 70, 120\}$

Question : sachant que $1 \leq N \leq 40$ et $50 \leq n_i \leq 1000$,
il est demandé de calculer la somme des poids des paniers d'au moins 200g.

Pour l'ex., les paniers intéressants sont $\{50, 60, 120\}, \{50, 70, 120\}, \{60, 70, 120\}, \{50, 60, 70, 120\}$
qui ont un poids total de $230 + 240 + 250 + 300 = 1020$

Exemple Kattis 1 : Fruit Baskets - Implémentation

Données : un entier N et une série d'entiers n_i, avec $i = 1, ..., N$.
Question : sachant que $1 \leq N \leq 40$ et $50 \leq n_i \leq 1000$,
il est demandé de calculer la somme des poids des paniers d'au moins 200g.

- Mauvaise solution :
  - Générer tous les paniers à l'aide de `itertools`
  - Ne pas prendre en compte ceux de poids < 200
  - Sommer les poids de ceux ≥ 200

  Comme N peut valoir 40, la complexité d'une telle approche est ingérable.

- Bonne solution :
  - Se rendre compte que, dans les paniers suivants

$\{50\}, \{60\}, \{70\}, \{120\}, \{50, 60\}, \{50, 70\}, \{50, 120\}, \{60, 70\}, \{60, 120\}, \{70, 120\}, \{50, 60, 70\}, \{50, 60, 120\}, \{50, 70, 120\}, \{60, 70, 120\}, \{50, 60, 70, 120\}$

50 apparait 8 fois, 60 apparait 8 fois, 70 apparait 8 fois et 120 apparait 8 fois.

  - La somme de tous les paniers est donc : $2^{N-1} \times \sum_i n_i$
  - Comme les fruits ont tous un poids ≥ 50g, il est envisageable de calculer la somme des paniers dont le poids est < 200g.
  - Effectuer la différence.

Exemple Kattis 2 : Perfect P*th* Powers - Description

Problème : https://open.kattis.com/problems/perfectpowers, de difficulté 5.0.

Données : une suite d'entiers se terminant par 0.

exemple : $n = [17 \quad 1073741824 \quad 25 \quad 0]$

Description du problème :
Pour chaque entier n (le 0 indique la fin du fichier),
on souhaite trouver le <u>plus grand entier p</u> tel que $b^p = n$ avec b entier.

Pour l'exemple :

- étant donné $n = 17$, l'output est **1** car $17^1 = 17$
- étant donné $n = 1073741824$, l'output est **30** car $2^{30} = 1073741824$
- étant donné $n = 25$, l'output est **2** car $5^2 = 25$

Exemple Kattis 2 : Perfect P*th* Powers - Implémentation

Description du problème :
Pour chaque entier n (le 0 indique la fin du fichier),
on souhaite trouver le plus grand entier p tel que $b^p = n$ avec b entier.

- A remarquer :
  - le b doit être un entier,
  - **donc b est un diviseur de n**
- Idée d'algorithme
  - pour chaque diviseur x de n
  - on regarde s'il est possible d'avoir l'égalité $x^p = n$ avec p entier
  - si on traite les diviseurs par ordre croissant, la première fois que l'égalité est vérifiée est la bonne!
  - si ce n'est jamais vérifié, c'est que $p = 1$

Oui, mais...

- n peut être négatif
- dans ce cas là, seules les valeurs de p impaires doivent être considérées