SÉANCE 4
MATHÉMATIQUES 2

Mattéo Delabre, Guillaume Pérution-Kihli & Julien Rodriguez

Université de Montpellier

21 février 2021

PLAN

FACTEURS PREMIERS

Tout entier strictement positif possède une unique décomposition en facteurs premiers. Soit n un entier et $i = 2$:

- Tant que $i^2 \leq n$
  1. Si $n \; mod \; i > 0$ alors $i = i + 1$
  2. Sinon $n = \frac{n}{i}$

- Si $n > 1$ alors on ajoute n dans la liste
- Retourner les facteurs de n

En *python3*, cet algorithme prend 0.216 *ms* pour $n = 600851475143$.

FACTEURS PREMIERS : AMÉLIORATION

Soit n un entier et $i = 2$:

- Tant que $i = 2$
  1. Si $n \ mod \ i > 0$ alors $i = i + 1$
  2. Sinon $n = \frac{n}{i}$
- Tant que $i^2 \leq n$
  1. Si $n \ mod \ i > 0$ alors $i = i + 2$
  2. Sinon $n = \frac{n}{i}$
- Si $n > 1$ alors on ajoute n dans la liste
- Retourner les facteurs de n

En *python3*, cet algorithme prend 0.106 *ms* pour $n = 600851475143$.

FACTEURS NON PREMIERS

Dans la plupart des concours de programmation il est généralement demandé de trouver la décomposition en facteur premier d'un nombre entier. Pour ce problème, nous allons compter le nombre de facteurs non premier d'un nombre.

Entrée *Ligne 1 : Q* Le nombre d'entiers à traiter. *n lignes suivantes :* Les entiers à traiter.

Limites $Q \leq 3.10^6$ et les entiers $n \leq 2.10^6$. Temps : 1 s. Mémoire : 1024 MB.

Sortie Un entier correspondant au nombre de facteurs non premiers pour chaque entier donné en entrée du problème.

Non-Prime Factors, ICPC 2018 ASR

PLAN

1. Facteurs premiers
2. **Principe d’inclusion/exclusion**
3. Exponentiation rapide modulaire
4. Combinatoire

PRINCIPE D'INCLUSION/EXCLUSION

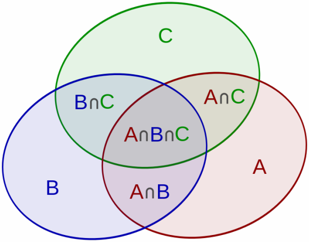
- Le principe d'inclusion-exclusion permet d'exprimer le cardinal d'une réunion finie d'ensembles finis en fonction du nombre d'éléments de ces ensembles et de leurs intersections

PRINCIPE D'INCLUSION/EXCLUSION

- Le principe d'inclusion-exclusion permet d'exprimer le cardinal d'une réunion finie d'ensembles finis en fonction du nombre d'éléments de ces ensembles et de leurs intersections
- $|A \cup B| = |A| + |B| - |A \cap B|$

PRINCIPE D'INCLUSION/EXCLUSION

- Le principe d'inclusion-exclusion permet d'exprimer le cardinal d'une réunion finie d'ensembles finis en fonction du nombre d'éléments de ces ensembles et de leurs intersections
- $|A \cup B| = |A| + |B| - |A \cap B|$
- $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$



PRINCIPE D'INCLUSION/EXCLUSION

- Le principe d'inclusion-exclusion permet d'exprimer le cardinal d'une réunion finie d'ensembles finis en fonction du nombre d'éléments de ces ensembles et de leurs intersections
- $|A \cup B| = |A| + |B| - |A \cap B|$
- $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$
- Généralisation : $|\bigcup_{i=1}^{n} A_i| = \sum_{k=1}^{n} (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \ldots < i_k \leq n} |A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_k}|$

PRINCIPE D'INCLUSION/EXCLUSION

- Le principe d'inclusion-exclusion permet d'exprimer le cardinal d'une réunion finie d'ensembles finis en fonction du nombre d'éléments de ces ensembles et de leurs intersections
- $|A \cup B| = |A| + |B| - |A \cap B|$
- $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$
- Généralisation : $|\bigcup_{i=1}^{n} A_i| = \sum_{k=1}^{n} (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \ldots < i_k \leq n} |A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_k}|$
- Variante : $\left|\bigcap_{i=1}^{n} \bar{A}_i\right| = |S - \bigcup_{i=1}^{n} A_i| = |S| - \sum_{i=1}^{n} |A_i| + \sum_{1 \leqslant i < j \leqslant n} |A_i \cap A_j| - \cdots + (-1)^n |A_1 \cap \cdots \cap A_n|$ avec $\forall i, A_i \subset S$ et $\bar{A}_i$ complémentaire de A_i dans S.

principe d'inclusion/exclusion : exercice

Exercice : Co-Prime

Indices :

- Calculer les facteurs premiers de N
- On peut calculer le nombre de nombres premiers avec N dans l'intervalle $[A, B]$ en calculant ceux dans les intervalles $[1, A-1]$ et $[1, B]$ puis en faisant la différence
- Pour calculer le nombre de nombre premiers avec N dans un intervalle, on peut calculer ceux qui ne le sont pas et faire la différence avec la taille de l'intervalle

PRINCIPE D'INCLUSION/EXCLUSION : EXERCICE

Exercice : Co-Prime

Indices :

- Calculer les facteurs premiers de N
- On peut calculer le nombre de nombres premiers avec N dans l'intervalle $[A, B]$ en calculant ceux dans les intervalles $[1, A-1]$ et $[1, B]$ puis en faisant la différence
- Pour calculer le nombre de nombre premiers avec N dans un intervalle, on peut calculer ceux qui ne le sont pas et faire la différence avec la taille de l'intervalle

Plus de détails et de problèmes : [Tutorial] Inclusion-Exclusion Principle

PLAN

EXPONENTIATION RAPIDE MODULAIRE

Soit deux entiers a et b, on souhaite calculer a^b.

- Il existe un algorithme en $O(log(b))$: la fonction **pow** de python

EXPONENTIATION RAPIDE MODULAIRE

Soit deux entiers a et b, on souhaite calculer a^b.

- Il existe un algorithme en $O(log(b))$: la fonction **pow** de python
- Cet algorithme utilise la relation :

$$a^{2^k}.a^{2^k} = a^{2^{k+1}}$$

EXPONENTIATION RAPIDE MODULAIRE

Soit deux entiers a et b, on souhaite calculer a^b.

- Il existe un algorithme en $O(log(b))$: la fonction **pow** de python
- Cet algorithme utilise la relation :

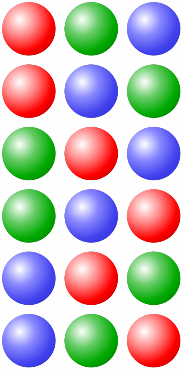
$$a^{2^k}.a^{2^k} = a^{2^{k+1}}$$

- Si le résultat est très grand, le résultat attendu peut être sous la forme : $a^b \mod q$, qui s'écrit en python comme ceci : **pow(a, b, q)**

PLAN

PERMUTATIONS

Soit trois boules, rouge verte et bleu. Combien de permutation pouvons nous former ?



PERMUTATIONS

Soit un ensemble de n objets :

- Le nombre de permutation est égal à $n!$, soit en python : **math.factorial(n)**

PERMUTATIONS

Soit un ensemble de n objets :

- Le nombre de permutation est égal à $n!$, soit en python : **math.factorial(n)**
- Si il est attendu toutes les permutations : **itertools.permutations(list)**

COMBINAISONS

Soit n le nombre d'objets et k la taille du sous ensemble. Le formule générale est :

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

- $prod = 1$
- pour i de 0 à k
  1. $prod = \frac{(prod \ . \ (n-i))}{i+1}$
- Retourner *prod*

Cet algorithme peut être facilement étendu pour un résultat modulo p premier (cf p. 180 *Programmation efficace*).

COMBINAISONS

Soit $A = \{a, b, c, d\}$, un ensemble de 4 éléments. Le nombre de combinaison de 2 éléments parmi les 4 éléments dans A est défini par :

$$\binom{4}{2} = \frac{4!}{2!(4-2)!} = 6$$

$$\{(a, b), (a, c), (a, d), (b, c), (b, d), (c, d)\}$$

Avec *itertools* : **itertools.combinations(A, k)**.

ARRANGEMENTS

Soit un ensemble de n éléments. Le nombre d'arrangements de k éléments parmi les n éléments dans B est défini par :

$$A_k^n = \frac{n!}{(n-k)!}$$

Avec **math.perm(n, k)**.

ARRANGEMENTS

Soit $B = \{a, b, c\}$, un ensemble de 3 éléments. Le nombre d'arrangements de 2 éléments parmi les 3 éléments dans B est défini par :

$$A_2^3 \;=\; \frac{3!}{(3-2)!} \;=\; 6$$

$$\{(a,b),(b,a),(a,c),(c,a),(b,c),(c,b)\}$$

Avec *itertools* : **itertools.permutation(B, k)**.