Matematika Diskret Dasar

Tim Olimpiade Komputer Indonesia

Pendahuluan

Melalui dokumen ini, kalian akan:

- Mempelajari **aritmetika modular**.
- Mempelajari bilangan prima.
- Memahami FPB dan KPK.
- Mempelajari algoritma **prime generation**.
- Mempelajari dan memanfaatkan **pigeon hole principle**.

Matematika Diskret

- Merupakan cabang matematika yang mempelajari tentang logika, sifat bilangan, kombinatorik, graf, dan lainnya.
- Diskret sendiri bermakna tidak kontinu, atau tidak bersinambungan.

Bagian 1

Arimetika Modular

Konsep Modulo

- Operasi a mod m biasa disebut "a modulo m".
- Operasi modulo ini akan memberikan sisa hasil bagi a oleh m.
- Contoh:
  - $5 \bmod 3 = 2$
  - $10 \bmod 2 = 0$
  - $21 \bmod 6 = 3$

Sifat-Sifat Modulo

- $(A + B) \bmod M = ((A \bmod M) + (B \bmod M)) \bmod M$
- $(A - B) \bmod M = ((A \bmod M) - (B \bmod M)) \bmod M$
- $(AB) \bmod M = ((A \bmod M) \times (B \bmod M)) \bmod M$
- $(A^B) \bmod M = ((A \bmod M)^B) \bmod M$
- $(A \times B) \bmod (A \times M) = A \times (B \bmod M)$

Aplikasi Modulo

- Pada pemrograman kompetitif, tidak jarang kita harus menghitung $n!$ mod k (terutama dalam kombinatorik).
- Seandainya kita menghitung $n!$, kemudian menggunakan operasi mod, kemungkinan besar kita akan mendapatkan integer overflow.
- Untungnya, kita dapat memanfaatkan sifat modulo.

Aplikasi Modulo (lanj.)

Solusi:

```
MODULARFACTORIAL(n, k)
1   result = 1
2   for i = 1 to n
3       result = (result × i) mod k
4   return result
```

Hati-Hati!

- Aritmetika modular tidak langsung bekerja pada pembagian.
- $\frac{a}{b} \mod n \neq \left(\frac{a \bmod n}{b \bmod n}\right) \mod n$.
- Contoh: $\frac{12}{4} \mod 6 \neq \left(\frac{12 \bmod 6}{4 \bmod 6}\right) \mod 6$.

- Pada aritmetika modular, $\frac{a}{b} \mod n$ biasa ditulis sebagai:

$$a \times b^{-1} \mod n$$

  dengan b^{-1} adalah **Modular Multiplicative Inverse** dari b.
- Jika tertarik, Anda bisa mempelajari *Modular Multiplicative Inverse* melalui link wikipedia ini.

Bagian 2

Bilangan Prima

Konsep Bilangan Prima

Bilangan Prima

Bilangan bulat positif yang hanya habis dibagi oleh 1 dan dirinya sendiri.
Contoh: $2, 3, 5, 13, 97$.

Bilangan Komposit

Bilangan yang memiliki lebih dari dua faktor.
Contoh: $6, 14, 20, 25$.

Primality Testing

- **Primality Testing** adalah algoritma untuk mengecek apakah suatu bilangan bulat N adalah bilangan prima.
- Kita dapat memanfaatkan sifat bilangan prima yang disebut di slide sebelumnya untuk mengecek apakah suatu bilangan merupakan suatu bilangan prima.

Primality Testing (lanj.)

- Solusi yang mudah untuk mengecek apakah N prima atau tidak tentu dengan mengecek apakah ada bilangan selain 1 dan N yang habis membagi N.
- Maka, kita dapat melakukan iterasi dari 2 sampai $N - 1$ untuk mengetahui apakah ada bilangan selain 1 dan N yang habis membagi N.
- Kompleksitas: $O(N)$.

Primality Testing (lanj.)

```
isPrimeNaive(n)
1   result = true
2   for i = 2 to n − 1
3       if n mod i == 0
4           result = false

5   if n < 2
6       result = false

7   return result
```

Primality Testing (lanj.)

- Ada solusi yang lebih cepat dari $O(N)$.
- Manfaatkan observasi bahwa jika $N = a \times b$, dan $a \leq b$, maka $a \leq \sqrt{N}$ dan $b \geq \sqrt{N}$.
- Kita tidak perlu memeriksa b, seandainya N habis dibagi b, tentu N habis dibagi a.
- Jadi kita hanya perlu memeriksa hingga $\sqrt{N}$.
- Kompleksitas: $O(\sqrt{N})$

Primality Testing (lanj.)

```
isPrimeSqrt(n)
1   result = true
2   i = 2
3   while i × i ≤ n
4         if n mod i == 0
5               result = false
6         i = i + 1

7   if n < 2
8         result = false

9   return result
```

Bagian 3

Generating Prime

Solusi Awal

- Kita dapat membangkitkan bilangan prima dengan iterasi dan *Primality Testing*.
- Solusi:
  1. Lakukan iterasi dari 2 sampai N
  2. Untuk tiap bilangan, cek apakah dia bilangan prima atau bukan. Jika iya, kita bisa memasukkannya ke daftar bilangan prima.

Solusi Awal (lanj.)

```
SIMPLEPRIMEGENERATION(n)
1   primeList = {}
2   for i = 2 to n
3       if ISPRIMESQRT(I)
4           primeList = primeList ∪ {i}
5   return primeList
```

Sieve of Erathostenes

- Terdapat solusi yang lebih cepat untuk membangkitkan bilangan prima, yaitu **Sieve of Erathostenes**.
- Ide utama utama dari algoritma ini adalah mengeliminasi bilangan-bilangan dari calon bilangan prima.
- Yang akan kita eliminasi adalah bilangan komposit.

Sieve of Erathostenes (lanj.)

- Kita tahu bahwa suatu bilangan komposit x dapat dinyatakan sebagai $x = p \times q$, dengan p suatu bilangan prima.
- Seandainya kita mengetahui suatu bilangan prima, kita dapat mengeliminasi kelipatan bilangan tersebut dari calon bilangan prima.
- Contoh: jika diketahui 7 adalah bilangan prima, maka 14, 21, 28, ... dieleminasi dari calon bilangan prima.

Prosedur Sieve of Erathostenes

1. Awalnya seluruh bilangan dari 2 sampai N belum dieleminasi.
2. Lakukan iterasi dari 2 sampai N:
   - 2.1 Jika bilangan ini belum dieliminasi, artinya bilangan ini merupakan bilangan prima.
   - 2.2 Lakukan iterasi untuk mengeliminasi kelipatan bilangan tersebut.

Implementasi Sieve of Erathostenes

- Kita dapat menggunakan **array boolean** untuk menyimpan informasi apakah suatu bilangan telah tereleminasi.
- Jika kita ingin mencari bilangan prima yang $\leq N$, maka diperlukan memori sebesar $O(N)$.
- Melalui perhitungan matematis, kompleksitas waktu solusi ini $O(N \; log \; log \; N)$.

Implementasi Sieve of Erathostenes (lanj.)

sieveOfErathostenes(n)

```
1   // Siapkan array eleminated berukuran n
2   // Inisialisasi array eleminated dengan false
3   primeList = {}
4   eleminated[1] = false
5   for i = 2 to n
6         if not eleminated[i]
7               primeList = primeList ∪ {i}
8               j = i
9               while i × j ≤ n
10                    eleminated[i × j] = true
11                    j = j + i
12  return primeList
```

Bagian 4

FPB dan KPK

Faktorisasi Prima

- Ketika masih SD, kita pernah belajar memfaktorkan bilangan dengan pohon faktor.
- Melalui faktorisasi prima, kita dapat menyatakan suatu bilangan sebagai hasil kali perkalian faktor primanya.
- Contoh: $7875 = 3^2 \times 5^3 \times 7$.
- Faktorisasi prima ini dapat bermanfaat pada soal-soal matematika di *Competitive Programming*.

FPB dan KPK

- FPB dan KPK dapat dicari melalui faktorisasi prima.
- Untuk setiap bilangan prima, kita menggunakan pangkat terkecil untuk FPB dan pangkat terbesar untuk KPK.
- Contoh:
  - $4725 = 3^3 \times 5^2 \times 7$
  - $7875 = 3^2 \times 5^3 \times 7$
- Maka:
  - $FPB(4725, 7875) = 3^2 \times 5^2 \times 7 = 1525$
  - $KPK(4725, 7875) = 3^3 \times 5^3 \times 7 = 23625$

- Perhatikan bahwa $KPK(a, b) = \frac{a \times b}{FPB(a,b)}$.

Algoritma Euclid

- Untuk mencari FPB suatu bilangan, menggunakan pohon faktor cukup merepotkan.
- Kita perlu mencari faktor prima bilangan tersebut, dan jika faktor primanya besar, tentu akan menghabiskan banyak waktu.
- Terdapat algoritma yang dapat mencari $FPB(a, b)$ dalam $O(\log(\min(a, b)))$.
- Algoritma ini bernama Algoritma Euclid.

Algoritma Euclid (lanj.)

```
EUCLID(a, b)
1  if b == 0
2        return a
3  else
4        return EUCLID(b, a mod b)
```

Pembuktian?

- Sangat pendek!
- Anda dapat menghemat waktu pengetikkan kode dalam melakukan pencarian FPB dengan algoritma ini.
- Jika Anda tertarik dengan pembuktiannya, baca lebih lanjut di link wikipedia ini.

Bagian 5

Pigeon Hole Principle

Pigeon Hole Principle

- Konsep PHP adalah "Jika ada N burung dan M sangkar, dimana $M > N$, maka ada sangkar yang berisi setidaknya 2 ekor burung".
- Secara matematis, jika ada N burung dan M sangkar, maka ada sangkar yang berisi setidaknya $\left\lceil \frac{M}{N} \right\rceil$ ekor burung.

Pigeon Hole Principle (lanj.)

- Terkesan sederhana?
- Simak contoh aplikasi prinsip ini.

Contoh Soal PHP

- Pak Dengklek memiliki sebuah **array** A berisi N bilangan bulat non-negatif.
- Anda ditantang untuk memilih angka-angka dari **array**-nya yang jika dijumlahkan habis dibagi N.
- Angka di suatu indeks **array** tidak boleh dipilih lebih dari sekali.
- Apabila mungkin, cetak angka-angka yang Anda ambil.
- Apabila tidak mungkin, cetak "Tidak mungkin".
- Batasan
  - $1 \leq N \leq 10^5$
  - **Array** berisi bilangan bulat non-negatif.

Analisis Contoh Soal PHP

- Inti permasalahan ini adalah mencari apakah pada **array** berukuran N, terdapat sub-himpunan tidak kosong yang jumlahan elemennya habis dibagi N.
- Jika ada, outputkan indeks-indeks yang terdapat di sub-himpunan tersebut.
- Jika tidak ada, keluarkan "Tidak Mungkin".

Analisis Contoh Soal PHP (lanj.)

- Mari kita coba mengerjakan versi lebih mudah dari soal ini: Bagaimana jika yang diminta sub-barisan, bukan sub-himpunan?
- Anggap array dimulai dari indeks 1 (*one-based*).
- Misalkan kita memiliki fungsi $sum(k) = \sum_{i=1}^{k} A[i]$.
- Untuk $sum(0)$, sesuai definisi nilainya adalah 0.

Analisis Contoh Soal PHP (lanj.)

- Kita dapat menyimpulkan bahwa $\sum_{i=l}^{r} A[i] = sum(r) - sum(l-1)$.
- Jika sub-barisan $A[l..r]$ habis dibagi N, maka $(sum(r) - sum(l-1)) \bmod N = 0$.
- Ini dapat kita tuliskan sebagai $sum(r) \bmod N = sum(l-1) \bmod N$.

Analisis Contoh Soal PHP (lanj.)

- Observasi 1:
  Ada N kemungkinan nilai $(sum(x) \bmod N)$, yaitu $[0..N-1]$.
- Observasi 2:
  Ada $N+1$ nilai x untuk $(sum(x) \bmod N)$, yaitu untuk $x \in [0..N]$.

  Ingat bahwa $sum(0)$ ada agar jumlahan sub-barisan $A[1..k]$ untuk tiap k dapat kita nyatakan dalam bentuk: $(sum(k) - sum(0))$.

Analisis Contoh Soal PHP (lanj.)

Observasi 3

- Ada $N + 1$ kemungkinan nilai x
- Ada N kemungkinan nilai $sum(x)$ mod N
- Pasti ada a dan b, sehingga $sum(b)$ mod $N = sum(a)$ mod N
- Sub-barisan yang menjadi solusi adalah $A[a + 1..b]$.

Analisis Contoh Soal PHP (lanj.)

- Dengan menyelesaikan versi mudah dari soal awal kita, ternyata kita justru dapat menyelesaikan soal tersebut.
- Suatu sub-barisan dari A pasti juga merupakan sub-himpunan dari A.
- Diketahui pula bahwa selalu ada cara untuk menjawab pertanyaan Pak Dengklek.

Implementasi

FINDDIVISIBLESUBSEQUENCE(A, N)

```
1   // Inisialisasi array sum[0..N] dengan 0
2   // Isikan nilai sum[i] dengan (A[1] + A[2] + ... + A[i])
3   // Inisialisasi array seenInIndex[0..N − 1] dengan −1

4   a = 0, b = 0
5   for i = 0 to N
6       if seenInIndex[sum[i] mod N] == −1
7           seenInIndex[sum[i] mod N] = i
8       else
9           a = seenInIndex[sum[i] mod N]
10          b = i

11  Print A[a..b]
```

Penutup

- Matematika diskret merupakan topik yang sangat luas.
- Materi ini berisikan beberapa konsep dasar matematika diskret yang umum digunakan pada pemrograman kompetitif.
- Selamat berlatih dan mengasah kemampuan!