BUKLET PEMBAHASAN SOAL

Penyisihan PROGRAMMING CONTEST ARKAVIDIA 4.0

6 Januari 2018

Soal-Soal

Kode	Judul	Penulis
A	Kue Ulang Tahun	Turfa Auliarachman
B	XOR	Alfonsus Raditya Arsadjaja
C	AND	Yonas Adiel Wiguna
D	Fahar, Jundi, dan Kotak	Yonas Adiel Wiguna
E	Sisa yang Dikuadratkan	Luqman Arifin
F	Hashing	Jauhar Arifin

A. Kue Ulang Tahun

Penulis soal	Turfa Auliarachman
Penulis editorial	Alfonsus Raditya Arsadjaja
Topik	*greedy*

Solusi

Misal besaran kue minimum adalah *min*. Menurut deskripsi soal, tiap anak harus makan besaran kue yang sama. Sisa kue menjadi paling mininum ketika tiap anak makan kue sebanyak mungkin. Karena ketersediaan kue, tiap anak paling banyak hanya bisa makan *min* kue. Karena itu, jawaban dari soal ini adalah jumlah dari semua potongan kue dikurangi *min* sebanyak jumlah anak (N), atau lebih formalnya:

$$\sum_{i=1}^{N} A_i - (N \times \min_{1 \le i \le N} A_i)$$

Kompleksitas solusi ini adalah $O(N)$.

B. XOR

Penulis soal	Alfonsus Raditya Arsadjaja
Penulis editorial	Luqman Arifin
Topik	*bit manipulation*

Solusi

Andaikan kita memiliki fungsi $f(x)$ yang menyatakan xor semua bilangan bulat dari 0 sampai x.
Atau secara formal:

$$f(x) = 0 \oplus 1 \oplus 2 \oplus ... \oplus x$$

Untuk x kelipatan 4, nilai $f(x)$ adalah x.
Untuk x sisa 1 jika dibagi 4, nilai $f(x)$ adalah 1.
Untuk x sisa 2 jika dibagi 4, nilai $f(x)$ adalah $x + 1$.
Untuk x sisa 3 jika dibagi 4, nilai $f(x)$ adalah 0.

Hasil xor semua bilangan dari L hingga R adalah $f(R) \oplus f(L - 1)$.
Kompleksitas solusi ini adalah $O(1)$.

C. AND

Penulis soal	Yonas Adiel Wiguna
Penulis editorial	Alfonsus Raditya Arsadjaja
Topik	*data structure*

Solusi

Soal ini cukup klasik. Bangun *segment tree*(`https://en.wikipedia.org/wiki/Segment_tree`) yang masing-masing node berisi nilai AND dari seluruh elemen pada range node tersebut.

Dengan cara yang sama, problem ini juga bisa diselesaikan menggunakan *sparse table*(`https://sites.google.com/site/indy256/algo/sparse_table_rmq`), atau *sqrt-decomposition* (`https://www.geeksforgeeks.org/sqrt-square-root-decomposition-technique-set-1-introduction/`).

Kompleksitas solusi ini adalah $O(N + Q \log N)$ atau $O(N \log N + Q)$ atau $O(N\sqrt{N} + Q\sqrt{N})$.

Alternatif Solusi

Pertimbangkan setiap bit secara independen. Untuk setiap *query*, anggap jawabannya adalah X. Bit ke-i pada X menyala apabila bit ke-i pada setiap elemen dalam rentang [L, R] juga menyala. Dengan kata lain, X memiliki bit ke-i nyala apabila banyak bit ke-i yang nyala dalam rentang [L, R] ada sebanyak $R - L + 1$.

Untuk dapat melakukan pengecekan dengan cepat, pertama-tama bangun *presum array* untuk kemunculan masing-masing bit. Apabila bit ke-i pada suatu elemen menyala, tandai dengan 1, selain itu tandai dengan 0. Lakukan hal ini sebelum *query* manapun.

Pada setiap *query*, pengecekan banyak bit ke-i yang nyala dalam rentang [L, R] dapat dilakukan dalam $O(1)$ dengan melihat nilai pref[i][R] - pref[i][L - 1].

Misalkan elemen maksimum A adalah MAX. Nilai MAX hanya sebesar 10^9, oleh karena itu untuk setiap *query* kita hanya perlu mempertimbangkan 30 bit pertama saja ($30 = \log 10^9$).

Kompleksitas solusi ini adalah $O(N \log MAX + Q \log MAX)$.

D. Fahar, Jundi, dan Kotak

Penulis soal	Yonas Adiel Wiguna
Penulis editorial	Luqman Arifin
Topik	*dynamic programming*

Catatan/Komentar

Kita perlu membagi array A menjadi beberapa *subset*. Tiap *subset* dicari selisih nilai maksimum dan minimumnya. Kita perlu meminimumkan jumlah dari selisih pada masing-masing subset.

Solusi

Nilai paling optimal terjadi ketika array A diurutkan, kemudian pemilihan subset terjadi paling optimal ketika kita memilih subsegmen dari A yang sudah diurutkan. Sesuai deskripsi soal, anggota elemen subsegmen minimal harus K.

Dengan ini, problem berubah menjadi: diberikan array A berisi N buah bilangan bulat. Partisi array tersebut menjadi beberapa subsegmen dengan banyak anggota subsegmen minimal K.

Asumsikan array berindeks dari 1 hingga N. Kita definisikan dp_x sebagai nilai minimum untuk mempartisi *prefix array* yang berakhir di x, dan x adalah ujung kanan dari subsegmen terakhir sementara. Jelas bahwa $dp_0 = 0$. Sesuai definisi, jawaban yang kita cari adalah dp_N.

Kita ingin dp_x seminimum mungkin. Misalkan $last$ adalah index terakhir subsegmen terbentuk tanpa melibatkan x. Dapat dilihat bahwa $dp_x = dp_{last} + hargaSubsegmen(last + 1, x)$ dengan syarat $x - last \geq k$. Untuk semua $last$ yang mungkin, kita ingin meminimumkan dp_x. Atau dengan kata lain:

$$dp_x = \min_{last \leq x-k} (dp_{last} + A_x - A_{last+1})$$

Karena A_x independen dari $last$, maka

$$dp_x = A_x + f(x)$$

dengan

$$f(x) = \min_{last \leq x-k} (dp_{last} - A_{last+1})$$

Dari sini dapat dilihat bahwa $f(x)$ berkorelasi dengan nilai dp di indeks sebelah kirinya. Nilai dp dapat dibangun dari kiri ke kanan secara iteratif. Misal nilai dp_{last} sudah diperoleh, kita perlu meng-update $f(last + k)$, $f(last + k + 1)$, $f(last + k + 2)$, ... $f(N)$.

Persoalan struktur data untuk menyimpan nilai $f(x)$ mirip dengan persoalan *range minimum query*. Kita butuh struktur data yang mampu update nilai minimum dalam range, dan mencari nilai minimum pada suatu elemen tunggal. Kita bisa gunakan *segment tree* atau BIT/Fenwick *tree*.

Jawabannya adalah $dp_N = A_N + f(N)$.

Kompleksitas solusi ini adalah $O(N \log N)$.

E. Sisa yang Dikuadratkan

Penulis soal	Luqman Arifin
Penulis editorial	Luqman Arifin
Topik	*math*

Catatan/Komentar

Melakukan operasi secara *naive* $O(T \cdot N \cdot Q)$ tentu saja akan mendapat *verdict timelimit*.

Solusi

Misal kita bangun array cnt_i yang menyatakan banyak kemunculan i pada array A. Dan misalkan bilangan terbesar pada A adalah MAX.

Misalkan pada saat ini kita akan menjawab pertanyaan untuk nilai X. Untuk setiap K kelipatan X, kita dapat menghitung kontribusi jawaban dari bilangan K, $K+1$, ..., $K+X-1$ dalam $O(1)$. Atau secara formal:

$$\sum_{i=K}^{K+X-1} cnt_i(i-K)^2$$

dapat dihitung dalam $O(1)$.

Menggunakan sifat $(a-b)^2 = a^2 - 2ab + b^2$, nilai di atas bisa diturunkan menjadi:

$$\left(\sum_{i=K}^{K+X-1} cnt_i \cdot i^2\right) - 2 \cdot K \left(\sum_{i=K}^{K+X-1} cnt_i \cdot i\right) + K^2 \left(\sum_{i=K}^{K+X-1} cnt_i\right)$$

bukti diserahkan kepada pembaca untuk latihan.

Untuk menghitung nilai di atas dalam $O(1)$ bisa digunakan 3 *prefix sum array* yang masing-masing menyimpan *presum* nilai kuadrat, *presum* nilai, dan *presum array cnt*.

Untuk setiap K yang merupakan kelipatan X, kompleksitasnya adalah $O(1)$. Untuk setiap X, perlu dipertimbangkan semua nilai K kelipatan X yang mungkin, sehingga kompleksitasnya adalah $O(\frac{MAX}{X})$. Untuk semua nilai X yang mungkin, kompleksitas total adalah:

$$O(\frac{MAX}{1} + \frac{MAX}{2} + \frac{MAX}{3} + ... + \frac{MAX}{MAX}) = O(MAX \log MAX)$$

Untuk X yang sama, kita tidak perlu menghitung jawaban dua kali. Untuk menghindari penghitungan ulang, kita bisa gunakan memoisasi/*cache*.

Kompleksitas solusi ini adalah $O(MAX \log MAX)$.

F. Hashing

Penulis soal	Jauhar Arifin
Penulis editorial	Luqman Arifin
Topik	*fast fourier transform*

Catatan/Komentar

Diberikan dua buah *array* A dan B. Untuk tiap *query* X, berapa banyak cara pengambilan satu elemen pada A dan satu buah elemen pada B sehingga jumlah kedua elemen tersebut sama dengan X?

Solusi

Kita bangun *array* P dengan P_i menyatakan banyak kemunculan i pada *array* A. Dengan cara yang sama, bangun juga *array* Q untuk *array* B.

Kita definisikan polinom $A(x)$ dan $B(x)$ sebagai berikut.

$$A(x) = P_{MAX} \cdot x^{MAX} + P_{MAX-1} \cdot x^{MAX-1} + \cdots + P_2 \cdot x^2 + P_1 \cdot x^1 + P_0 \cdot x^0$$

$$B(x) = Q_{MAX} \cdot x^{MAX} + Q_{MAX-1} \cdot x^{MAX-1} + \cdots + Q_2 \cdot x^2 + Q_1 \cdot x^1 + Q_0 \cdot x^0$$

$A(x)$ dan $B(x)$ adalah polinom berderajat $MAX \leq 50.000$. Apabila $A(x)$ dan $B(x)$ dikalikan, misal $C(x) = A(x) \cdot B(x)$, maka jawaban untuk *query* X adalah koefisien suku x^X dari polinom C.

Mengalikan polinom secara *naive* membutuhkan $O(MAX^2)$. Dengan *fast fourier transform*, mengalikan dua buah polinom bisa dilakukan dalam $O(MAX \log MAX)$.

Kompleksitas solusi ini adalah $O(MAX \log MAX)$.