Pembahasan Coder Class

SCPC – Minggu 1

Daftar Soal

A. Olimpiade Chanek

Tag

Matematika

Pembahasan

Mula-mula anak dan badak adalah himpunan yang terpisah sehingga jumlah populasi bisa didapatkan yaitu $A + B$. Setelah terjadi perubahan, jumlah anak bertambah dengan badak 'setengah anak' (mula-mula badak) dan jumlah badak bertambah dengan anak 'setengah badak' (mula-mula anak). Sehingga untuk mencari jumlah badak 'setengah anak' dan anak 'setengah badak' adalah dengan $C + D - (A + B)$.

Hati-hati karena hasil akhir bisa saja melebihi batas bilangan bulat 32-bit.

Kompleksitas: $O(1)$

B. Bilangan Kuadrat Terbesar

Tag

Matematika, Divide and conquer

Pembahasan

Misal jawaban yang ingin kita dapatkan adalah X. Ketimbang berpikir mencari X, sehingga X^2 merupakan bilangan kuadrat terbesar yang memiliki N digit pada basis B, lebih mudah untuk mencari $X + 1$, sehingga $(X + 1)^2$ merupakan bilangan kuadrat terkecil yang memiliki $N + 1$ digit pada basis B. Kenapa? Perhatikan bahwa akibat N bilangan genap, pada basis B apapun, pasti $(X + 1)^2$ memiliki bentuk

$$1\underbrace{000\ldots00000}_{N\ buah}$$

Maka, apakah $X + 1$? Ya, pasti $X + 1$ adalah $B^{N/2}$! Sekarang, kita sudah mempunyai X, yang berarti kita bisa mendapatkan X^2 dengan mudah.

Permasalahan sekarang ada 2, yaitu:

1. N bernilai sangat besar, sehingga menghitung $B^{N/2}$ tidak mudah
2. Terdapat batasan mencetak D digit terakhir

Untuk permasalahan pertama, kita dapat melakukan penghitungan dalam $O(\log N)$, menggunakan *fast modular exponentiation*.

Untuk permasalahan kedua, perhatikan bahwa mencari D digit terakhir dari suatu bilangan Y dalam basis B sama saja dengan hasil dari $Y\ mod\ (B^D)$, sehingga D digit terakhir dari X^2 adalah $X^2\ mod\ (B^D)$. Namun, terdapat masalah baru. Apabila $N < D$, maka kita hanya perlu mencetak X^2. Namun, apabila $N \geq D$, maka kita harus mencetak D digit terakhir, yang berarti dari $X^2\ mod\ (B^D)$ mungkin saja harus kita tambahkan dengan *leading zero*.

Kompleksitas: $O(\log N)$

C. Channek Pen sCanner

Tag

ad hoc

Pembahasan

Kita cukup memeriksa sesuai deskripsi soal. Pada saat memeriksa, kita bisa memeriksa beberapa karakter saja yang bisa membedakan ‘A’, ‘B’, ‘C’, dan spasi untuk mempercepat pemeriksaan.

Bagian yang cukup *tricky* pada soal ini adalah batasan 150.000 huruf. Ingat, 150.000 huruf berarti membutuhkan panjang 600.000 karakter. Hal ini bisa menjebak peserta yang menggunakan *array of char* untuk menyimpan string masukan.

Kompleksitas: $O(|input|)$

D. Pululu

Tag

Ad hoc

Pembahasan

Terdapat 2 kasus pada soal ini, yaitu ketika N atau M bernilai 1, dan ketika keduanya bernilai lebih dari 1.

Untuk kasus pertama, jelas kita bisa mengisi semua petak dengan pululu. Maka, pada kasus ini jawabannya adalah $N + M - 1$.

Untuk kasus kedua, salah satu pemasangan yang optimal adalah sebagai berikut: Awalnya, kita mengisi semua petak pada baris pertama dengan pululu. Lalu, kita mengisi semua petak pada kolom pertama dengan pululu. Saat ini, kita telah memasang $N + M - 1$ pululu. Perhatikan bahwa setelah pemasangan ini, kita masih bisa meletakkan 1 pululu lagi, yaitu di petak pada kolom dan baris terakhir. Maka, pada kasus ini jawabannya adalah $N + M$.

Kompleksitas: $O(1)$

E. String Seimbang

Tag

Ad hoc, Stack, Bracket Matching

Pembahasan

Pertama-tama, misal ‘(‘ kita anggap 1 dan ‘)’ kita anggap -1. Kemudian, misal Bal_i didefinisikan sebagai jumlahan nilai-nilai tersebut, dari 1 hingga i. Untuk kemudahan bersama, definisikan $Bal_0 = 0$. Maka, untuk suatu string seimbang yang dimulai dari karakter ke-L dan berakhir di karakter ke-R, pasti memiliki ciri berikut:

1. $Bal_R - Bal_{L-1} = 0$, intuitifnya setiap ‘(‘ pasti berpasangan dengan ‘)’
2. Untuk setiap $L \leq i \leq R, Bal_i - Bal_{L-1} \geq 0$, intuitifnya tidak ada ‘)’ yang tidak memiliki pasangan

Selanjutnya, misal $counter_i$ menyatakan banyaknya indeks k, sehingga $Bal_k = i$. untuk menghitung banyaknya substring seimbang, kita lakukan iterasi dari 1 sampai N, dan melakukan tindakan berikut (misal kita memproses indeks i):

1. Apabila karakter yang sedang diproses adalah ‘(‘, maka seluruh indeks k $(k < i)$ dengan nilai $Bal_k = Bal_i$ tidak mungkin dipasangkan dengan indeks m manapun $(m > i)$ yang memiliki nilai Bal yang sama, karena akan menyalahi ciri 2. Maka, kita ubah $counter_{Bal_i}$ menjadi 1

2. Apabila karakter yang sedang diproses adalah ‘)‘, maka seluruh indeks yang terhitung pada $counter_{Bal_i}$ dapat menjadi awalan substring seimbang yang berakhir di i. Maka, jawaban bertambah sebanyak $counter_{Bal_i}$, dan $counter_{Bal_i}$ bertambah 1.

Perhatikan bahwa pada awalnya, $counter_0$ = 1. Setelah iterasi berakhir, kita sudah mendapatkan jawaban yang kita inginkan.

Untuk referensi, Anda dapat membaca solusi resmi di sini

Kompleksitas: $O(N)$

F. Middle of Nowhere

Tag

Matematika

Pembahasan

Pertama-tama, ingat bahwa $a_1 + a_2 + \ldots + a_M = N$. Kita dapat memanfaatkan pertidaksamaan *Arithmatic Mean* dan *Harmonic Mean*, sebagai berikut:

$$AM \geq HM$$

$$\frac{a_1 + a_2 + \cdots + a_M}{M} \geq \frac{M}{\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_M}}$$

$$\frac{N}{M} \geq \frac{M}{\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_M}}$$

$$\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_M} \geq \frac{M^2}{N}$$

Dapat dilihat bahwa batas bawah untuk total kesialan adalah $\frac{M^2}{N}$. Sehingga, total kesialan minimum adalah $\frac{M^2}{N}$.

Referensi:

- https://artofproblemsolving.com/wiki/index.php?title=Root-Mean_Square-Arithmetic_Mean-Geometric_Mean-Harmonic_mean_Inequality

Kompleksitas: $O(1)$

G. Jalan-Jalan

Tag

Dynamic programming

Pembahasan

Misal S merupakan $\sum_{i=1}^{N} D_i$. Maka, $y = S - x$. Sehingga, kita ingin memaksimalkan nilai $x \cdot y = x \cdot (S - x)$. Dengan sedikit observasi tambahan, kita tahu bahwa $x \cdot (S - x)$. semakin besar apabila selisih x dengan $S - x$ semakin kecil. Oleh karena itu, sekarang kita ingin mencari tahu apakah untuk suatu x pada $[0, S]$, kita dapat membentuknya sebagai jumlahan beberapa D_i. Setelah itu, untuk setiap x yang dapat dibentuk, kita cari yang memaksimalkan $x \cdot (S - x)$.

Mencari tahu apakah untuk suatu x pada $[0, S]$, kita dapat membentuknya sebagai jumlahan beberapa D_i merupakan salah satu permasalahan klasik, yaitu *subset sum*. Kita dapat menyelesaikannya dengan menggunakan *dynamic programming*, dengan *state* DP-nya berupa (*posisi*, *jumlah_yang_ingin_dibentuk*). Kurang lebih, formulasi DP-nya sebagai berikut:

$$f(pos, target) = \begin{cases} true & , pos = N \wedge target = 0 \\ false & , pos = N \wedge target \neq 0 \\ f(pos+1, target) & , pos < N \wedge target < D_{pos} \\ f(pos+1, target) \vee f(pos+1, target - D_{pos}) & , pos < N \wedge target \geq D_{pos} \end{cases}$$

Mencari tahu apakah suatu x dapat dibentuk dapat dilakukan dengan memanggil $f(0, x)$. Untuk mencari tahu pembagian kanan dan bawah, kita bisa melakukan *backtracking* dengan memanfaatkan tabel DP-nya.

Kompleksitas: $O(NS)$

H. Baskom Mania

Tag

Ad hoc

Pembahasan

Pertama-tama, terdapat kasus khusus untuk soal ini, yaitu ketika M = 0, yang mana jawabannya adalah 0 karena sejak awal bak mandinya sudah kosong.

Selanjutnya, kita definisikan *cycle_sum* sebagai $\sum_{i=1}^{N}(K - A_i)$. Kemudian, definisikan array S, dengan S_i sebagai $M + \sum_{j=1}^{i}(K - A_j)$. S dapat dihitung dalam $O(N)$.

Kemudian, kita lakukan iterasi, dari anak pertama hingga ke-N, berapa kali penggunaan gayung minimum sehingga anak ke-i tidak bisa mengambil air, atau setelah anak ke-i mengambil air bak mandi menjadi kosong, dengan asumsi apabila 2 hal tersebut terjadi pada anak yang lain, penggunaan gayung tetap dilanjutkan.

Ternyata, ada 4 kemungkinan kasus yang dapat terjadi:

1. $S_i = 0$. Maka, setelah anak ke-i mengambil air untuk pertama kalinya, bak mandi menjadi kosong. Terjadi i penggunaan gayung.
2. $S_i < 0$. Maka, anak ke-i tidak bisa mengambil air karena air pada bak mandi kurang. Terjadi $i - 1$ penggunaan gayung.

3. $S_i > 0$ dan *cycle_sum* < 0. Ini artinya, setelah sekian *cycle*, akan terjadi salah satu dari 2 kemungkinan kasus di atas. Banyaknya *cycle* yang terjadi adalah $\left\lceil \frac{S_i}{-cycle\_sum} \right\rceil$. Banyaknya penggunaan gayung pada kasus ini adalah $N * \left\lceil \frac{S_i}{-cycle\_sum} \right\rceil + i - 1$, kecuali apabila S_i habis dibagi *cycle_sum*, yang mana banyaknya penggunaan gayung adalah $N * \left\lceil \frac{S_i}{-cycle\_sum} \right\rceil + i$.
4. Selain itu, tidak mungkin penggunaan gayung berakhir pada anak ke-i.

Selanjutnya, jawaban yang kita cari adalah nilai minimum perhitungan di atas untuk setiap anak. Apabila kasus 4 terjadi untuk semua anak, maka jawabannya adalah -1.

Kompleksitas: $O(N)$