Pembahasan Penyisihan JCPC dan SCPC 12

Scientific Commitee

September 2020

Junior Competitive Programming Contest

A Bintang Ojol

- Penulis soal: Julian Fernando
- Tingkat kesulitan perkiraan: Mudah
- Tag: *Math, Greedy*

A.1 Subsoal 1

Jika $N = 1$, maka jumlah ojol yang mungkin untuk diberikan bintang adalah 1. Jadi, jika $M = 1$, cetak "0 0" dan jika $M > 1$, cetak "-1 -1".

Kompleksitas: $O(1)$

A.2 Subsoal 2

Jika $M = 1$, maka satu-satunya kemungkinan pemberian bintang adalah ojol tersebut mendapatkan semua bintang. Jika $N < 5$, cetak "0 0". Jika $N = 5$, cetak "1 1" dan jika $N > 5$, cetak "-1 -1".

Kompleksitas: $O(1)$

A.3 Subsoal 3

Kamu bisa menggunakan *Dynammic Programming* untuk menyelesaikan masalah ini untuk mencari nilai maksimum dan minimum. Jika mencari nilai maksimum, kamu misalkan $DP[i][j]$ sebagai jumlah 5 bintang maksimum yang mungkin didapat dengan memberi rating kepada ojol ke - i dengan sisa bintang sekarang j. Transisinya:

$$DP[i][j] = max_{1 \leq k \leq min(j,5)} \left(DP[i-1][j-k] + (k == 5)\right)$$

Jawaban yang kamu inginkan akan didapatkan jika kamu memanggil $DP[M][N]$. DP yang serupa dapat kamu gunakan untuk kasus nilai minimum.

Kompleksitas: $O(NM)$

A.4 Subsoal 4

Kamu bisa mencoba-coba setiap kemungkinan dari banyaknya ojol penerima 5 bintang. Misalkan jumlah penerima 5 bintang adalah X ojol dimana $1 \leq X \leq M$. Maka sisa bintang yang perlu didistribusikan adalah $N' = N - 5X'$ dan sisa ojol adalah $M' = M - X$. kamu bisa mendistribusikan bintang-bintang ini jika $M' \leq N' \leq 4M'$, yaitu kasus terkecil jika semuanya mendapatkan 1 bintang dan kasus terbesar jika semuanya mendapatkan 4 bintang.

Kompleksitas: $O(M)$

A.5 Subsoal 5

Kasus terkecil adalah semua ojol mendapatkan 1 bintang dan kasus terbesar adalah semua ojol mendapatkan 5 bintang. Jadi, jika $N < M$ atau $N > 5M$ maka cetak "-1 -1". Untuk mendapatkan nilai minimum, kamu perlu memberikan 4 bintang kepada semua ojol sehingga jika $N \leq 4M$, jawabannya adalah 0 dan jika $N > 4M$ jawabannya adalah $N - 4M$. Untuk nilai maksimum, Perhatikan bahwa setiap ojol harus diberikan paling sedikit 1 bintang sehingga kamu perlu memberikan masing-masing 1 bintang kemudian mencari berapa maksimum ojol yang dapat ditambahhkan 4 bintang. Jadi, jawaban nilai maksimum adalah $\lfloor (N - M)/4 \rfloor$

A.6 Kode sumber

https://ideone.com/UlHen5

B Membuat Elemen

- Penulis soal: Hocky Yudhiono, Julian Fernando
- Tingkat kesulitan perkiraan: Mudah
- Tag: *Math, Brute Force*

B.1 Subsoal 1

Kamu bisa mencoba setiap kemungkinan nilai untuk A, B, dan C.

Kompleksitas: $O(N^3)$

B.2 Subsoal 2

Kamu bisa mencoba setiap kemungkinan nilai untuk B dan C. Nilai A dapat dihitung dengan $A = N - BC$. Sehingga percobaan untuk setiap kemungkinan dapat dilakukan lebih cepat.

Kompleksitas: $O(N^2)$

B.3 Subsoal 3

Kamu bisa mencoba setiap kemungkinan nilai untuk A. Untuk setiap A, sebut $N' = N - A$. Kamu bisa mencari semua faktor dari N' dalam $O(\sqrt{N})$ lalu validasi tripel (A, B, C) ini (apakah berbeda satu sama lain)

Kompleksitas: $O(N\sqrt{N})$

B.4 Subsoal 4 dan 5

Kamu bisa menggunakan algoritma yang sama pada Subsoal 2 tetapi pada saat $BC > N$ lakukan break pada program. Hal ini akan membuat kompleksitas turun menjadi $O(N \log N)$. Perhitungannya: Untuk suatu nilai $B = X$, kamu hanya mengecek nilai C dari 1 hingga $\lfloor N/X \rfloor$ inklusif. Jadi, dengan menggunakan perkiraan *Euler*, total kemungkinan yang kamu cek adalah:

$$\frac{N}{1} + \frac{N}{2} + \frac{N}{3} + ... + \frac{N}{N} \approx NlnN$$

Kompleksitas: $O(N \log N)$

B.5 Kode Sumber

https://ideone.com/sh6el7

C Energi Kandang

- Penulis soal: Galangkangin Gotera
- Tingkat kesulitan perkiraan: Sulit
- Tag: *Dynamic Programming, Deque*

C.1 Subsoal 1

Karena N = R, pasti ada kemungkinan dimana seluruh kandang digabung menjadi 1 kandang. Pada kasus ini hanya ada 1 kemungkinan energi, sehingga jawabannya pasti **tidak lebih besar** dari rata-rata dari seluruh A_i. Oleh karena ini, untuk kasus ini pasti $K = 1$. Tetapi kamu bisa melihat bahwa jawaban optimalnya pasti saat X = rata-rata tersebut. Jika misalkan ada kemungkinan partisi lain dimana energi dari setiap partisinya kurang dari X, maka rata-rata totalnya pasti juga kurang dari X sehingga kamu dapati bahwa untuk kemungkinan partisi-partisi lain pasti ada setidaknya satu kandang yang energinya $\geq X$.

Kompleksitas: $O(N)$

C.2 Subsoal 2

Karena K = 1, pada suatu partisi tentunya optimal untuk memilih X sebagai energi terbesar (rata-rata terbesar) di partisi tersebut. Oleh karena itu permasalahan ini dapat dimodelkan dalam $DP[i]$ = rata-rata terbesar paling kecil jika ingin membagi kandang 1..i kedalam partisi- partisi. Transisinya:

$$DP[i] = min_{i-R \leq j \leq i-1} \left(max(DP[j], avg(j+1, i)\right)$$

Dimana $avg(j, i)$ menyatakan rata-rata dari semua kandang pada kandang j hingga i inklusif.

DP Diatas memiliki N transisi. Nilai avg(j, i) dapat dihitung secara naif dalam $O(N)$. Sehingga total kompleksitasnya $O(N^2)$. Jawabannya kemudian $DP[N]$

C.3 Subsoal 3

kamu bisa binary search the answer. Misalkan kamu bisa memilih pakan dengan energi X, pasti kamu bisa memilih pakan dengan energi $\leq X$. Jika kamu tidak bisa memilih pakan dengan energi X, pasti tidak bisa memilih pakan dengan energi $\geq X$. Bagaimana caranya cek untuk suatu X?

Anggap kamu mau cek dengan cepat apakah $avg(i, j) \geq X$. Dengan menguraikan:

$$avg(i, j) = \frac{A_i + A_{i+1} + A_{i+2} + ... + A_j}{j - i + 1} \geq X$$

$$A_i + A_{i+1} + A_{i+2} + ... + A_j \geq X(j - i + 1)$$
$$(A_i - X) + (A_{i+1} - X) + (A_{i+2} - X) + ... + (A_j - X) \geq 0$$

Perhatikan ruas kiri terakhir bersifat independen terhadap A dan X. Sehingga jika di awal kamu membuat array baru $A', A'[i] = A[i] - X$ lalu membuat prefix sum P, kamu dapat menghitung apakah $avg(i, j) \geq X$ dalam O(1) yakni $P[j] - P[i - 1] \geq 0$. Didefinisikan $avg2(i, j) = 1$ apabila $avg(i, j) \geq X$ dan 0 jika tidak. Karena sedang melakukan binary search, kamu hanya tertarik untuk suatu X apakah untuk setiap kemungkinan partisi apakah ada K atau lebih elemen yang rata-ratanya tidak kurang dari X. kamu bisa mendefinisikan $DP[i]$ = banyaknya partisi minimal yang rata-ratanya $\geq X$ dari semua kemungkinan mempartisi array dari 1..i. Transisinya:

$$DP[i] = min_{i-R \leq j \leq i-1} (DP[j] + avg2(i + 1, j))$$

Jika $DP[N] \geq K$, maka pada setiap partisi minimal ada K bagian yang rata-ratanya $\geq X$. DP diatas dapat dihitung dalam $O(N^2)$.

Kompleksitas: $O(N^2 \log N)$

C.4 Subsoal 5

Saat menghitung DP pada subsoal 3, perhatikan bahwa nilai $0 \leq avg2(i+1, j) \leq 1$. Anggap $U = min_{i-R\ lej \leq i-1} (DP[j])$. Didapatkan observasi $DP[i]$ hanya mungkin bernilai U atau $U + 1$. Jadi kamu cek apakah mungkin $DP[i] = U$. Hal ini mungkin apabila ada j yang valid dimana $DP[j] = U$ dan $P[j] > P[i]$. Perhatikan bahwa hal ini tidak bergantung pada i. Sehingga kamu untuk setiap kemungkinan nilai DP, kamu bisa menyimpan dalam array of multiset indeks peserta nilai prefix dari indeks tersebut, i.e: $M[x]$ menyatakan multiset dari semua indeks j dimana $DP[j] = x$. Jadi untuk suatu i dan U, kamu cari prefix terbesar pada $M[U]$ yang masih valid lalu cek apakah nilainya lebih besar dari $P[i]$. Jika nilai $DP[i]$ tidak mungkin U, nilainya pasti $U + 1$.

code: https://ideone.com/4Eu4Cn

Kompleksitas: $O(N \log^2 N)$

C.5 Subsoal 4

kamu melakukan binary search dan DP seperti pada subsoal 3 . Ada observasi tambahan yang dapat dimanfaatkan. Perhatikan bahwa jika untuk suatu i, $DP[i] > 1$, kamu bisa set $DP[i] = 1$ saja karena $K = 1$ sehingga nilai DP yang lebih besar tidak akan berpengaruh. Jadi kamu dapati nilai DP hanya mungkin 0 atau 1. Perhatikan kembali persamaan DP pada subsoal 3. kamu mau meminimalisir nilai dari semua indeks yang mungkin dan perhatikan juga nilai $avg2$ maksimal 1.

Sehingga, untuk setiap i, kamu cek apakah mungkin mendapatkan $DP[i] = 0$. Hal ini hanya mungkin terjadi jika ada j valid yang memenuhi $DP[j] = 0$ dan $P[j] > P[i]$. Sehingga kamu bisa menyimpan nilai $P[j]$ untuk setiap i dimana

$DP[j] = 0$. Menyimpan DP dengan set seperti pada subsoal 5 akan menambah faktor $\log N$ sehingga harus mencari cara lain untuk menyimpan jawaban.

Ada observasi bahwa untuk suatu j dan k, jika $DP[j] = DP[k] = 0$, $k > j$, $P[k] > P[j]$ pada pasti lebih optimal untuk memilih k dibanding j karena kemungkinan mendapati $P[k] > P[i]$ lebih besar. Sehingga, kamu bisa menyimpan sebuah deque berisi indeks-indeks yang masih mungkin dipilih. jika ada indeks j dan k pada deque dimana $j < k$, pasti $P[j] > P[k]$. Jadi kamu bisa mengupdate deque sebagai berikut:

1. Saat menaikkan nilai i, hapus semua elemen dari belakang deque yang dimana indeksnya sudah tidak valid (berada diluar range $[L, R]$
2. hapus semua indeks j dari depan deque yang memenuhi $P[j] \leq P[i]$ jika $DP[i] = 0$.

Dengan demikian, indeks valid pasti berada di depan deque. Dapat diperhatikan bahwa setiap indeks hanya akan dilewati sekali sehingga kompleksitas total algoritma deque $O(N)$.

Kompleksitas: $O(N \log N)$

C.6 Subsoal 6

kamu bisa mengembangkan algoritma deque pada subsoal 4 dan menggabungkan observasi U pada subsoal 5. Sekarang deque tidak hanya menyimpan nilai prefix, tetapi juga menyimpan nilai DP pada indeks tersebut. Jadi, dua indeks j dan k, $j < k$ berada di deque jika:

- $DP[j] > DP[k]$
- $DP[j] = DP[k], P[j] > P[k]$

Selanjutnya untuk menjaga deque mirip seperti subsoal 4. Dari depan deque hapus indeks yang sudah tidak valid (keluar jangkauan), dari belakang deque coba masukkan indeks i dan hapus elemen yang menjadi tidak valid. Selanjutnya saat menghitung DP, elemen optimal pasti berada di depan deque.

Kompleksitas: $O(N \log N)$

C.7 Kode Sumber

https://ideone.com/n20L4F

D Duel Maut

- Penulis soal: Julian Fernando
- Tingkat kesulitan perkiraan: Menengah-Sulit
- Tag: *Two Pointer, Data Structure*
- **Fun Fact** 1: Mencetak 0 untuk setiap query mendapatkan 8 poin dari subsoal 9 :).

Inti dari soal ini adalah mencari banyak indeks i sehingga ($A_i < X_j$ dan $B_i > Y_j$) atau ($A_i > X_j$ dan $B_i < Y_j$).

D.1 Subsoal 1

Pada subsoal ini, batasan untuk nilai N dan Q cukup kecil sehingga setiap *query* dapat dilakukan pengecekan untuk semua kekuatan kartu dan dibandingkan dengan stamina yang dimiliki.

Kompleksitas: $O(NQ)$

D.2 Subsoal 2

Pada subsoal ini, nilai untuk kekuatan kartu hanya berada pada rentang -10 sampai 10 inklusif. Kamu dapat membuat *array* 2D $Cnt_{x,y}$ yang berisi banyaknya indeks i dimana $A_i = x$ dan $B_i = y$. Jadi, untuk setiap *query*, kamu dapat mengecek dari array tersebut.

Kompleksitas: $O(21 * 21 * Q)$

D.3 Subsoal 3

Pada subsoal ini, hanya nilai A_i yang berada pada rentang -10 sampai 10 inklusif. Kamu bisa membuat vector untuk menyimpan nilai B_i dari setiap nilai A_i. Setelah semua vector tersebut diurutkan, kamu dapat mengecek semua nilai A_i dan melakukan *binary search* untuk nilai B_i.

Kompleksitas: $O(21 * Q * log_2(N))$

D.4 Subsoal 4

Pada subsoal ini, perlu dilakukan *compress* untuk memperkecil ukuran. Perhatikan bahwa nilai A_i dan B_i berada pada rentang -31 sampai 31 inklusif. Jadi, jika nilai X_j atau Y_j lebih kecil dari -31, dapat kamu ubah menjadi -32 dan jika lebih besar dari 31, dapat kamu ubah menjadi 32. Jadi, hanya ada 65 kemungkinan nilai X_j dan 65 kemungkinan nilai Y_j. kamu dapat menggunakan algoritma *brute force* dan menyimpan jawabannya di *array*.

Kompleksitas: $O(65 * 65 * N)$

D.5 Subsoal 5

Pada subsoal ini, karena semua A_i nilainya sama maka kamu dapat mengurutkan nilai B_i dan melakukan *binary search*.

Kompleksitas: $O(Q * log_2(N))$

D.6 Subsoal 6

Pada subsoal ini, karena semua Y_j nilainya sama maka kamu dapat memisahkan kartu tersebut menjadi 2 kelompok yaitu $B_i < Y_j$ dan $B_i > Y_J$. Setelah itu, kamu dapat mengurutkannya nilai A_i kemudian melakukan *binary search*.

Kompleksitas: $O(Q * log_2(N))$

D.7 Subsoal 7

Pada subsoal ini, nilai $A_i = B_i$ sehingga kamu dapat menganggapnya sebagai suatu titik sedangkan nilai X_J dan Y_j dapat kamu anggap sebagai interval. Sekarang kamu hanya perlu mencari berapa titik yang terdapat di antara interval tersebut (eksklusif). Untuk melakukannya, kamu dapat mengurutkan posisi titik tersebut dan melakukan *binary search* sebanyak dua kali untuk mendapatkan batas atas dan batas bawahnya.

Kompleksitas: $O(Q * log_2(N))$

D.8 Subsoal 8

Subsoal ini mirip dengan Subsoal 7. Perbedaannya adalah nilai A_i dan B_i yang kamu anggap sebagai interval sehingga kamu perlu mencari ada berapa interval yang terdapat nilai X_j di dalamnya.

Kompleksitas: $O(Q * log_2(N))$

D.9 Subsoal 9

Pada subsoal ini, nilai $A_i = B_i$ dan nilai $X_j = Y_j$. Jadi, dari kamu perlu mencari banyaknya indeks i sehingga ($B_i < Y_j$ dan $B_i > Y_j$) atau ($B_i > Y_j$ dan $B_i < Y_j$). Tidak ada dua buah nilai P dan Q sehingga $P < Q$ sekaligus $P > Q$. Hal ini membuat semua jawaban adalah 0.

Kompleksitas: $O(Q)$

D.10 Subsoal 10

Pada subsoal ini, perlu dilakukan *compress* untuk memperkecil ukuran. Perhatikan bahwa nilai A_i dan B_i berada pada rentang -1937 sampai 1937 inklusif. Jadi, jika nilai X_j atau Y_j lebih kecil dari -1937, dapat kamu ubah menjadi -1938 dan jika lebih besar dari 1937, dapat kamu ubah menjadi 1938. Jadi, hanya ada 3875 kemungkinan nilai X_j dan 3875 kemungkinan nilai Y_j. kamu dapat *precompute* semua kemungkinan jawaban dengan 2 kali *for loop*. Pertama

untuk mencari ($A_i < X_j$ dan $B_i > Y_j$) dan yang kedua untuk mencari ($A_i > X_j$ dan $B_i < Y_j$).

Kompleksitas: $O(2 * 3875 * 3875)$

D.11 Subsoal 11

Pada subsoal ini, kamu perlu membuat dua buah kelompok dimana kelompok pertama memiliki atribut $A_i < X_j$ dan kelompok kedua memiliki atribut $A_i > X_j$. Untuk setiap *query*, kamu harus terus memperbaharui dua buah kelompok ini agar atributnya tetap berlaku. Hal ini dapat dilakukan dengan menggunakan **Segment Tree/BIT/PBDS**.

Kompleksitas: $O(Q * log_2(N))$

D.12 Subsoal 12

Pada subsoal ini, kamu dapat menggunakan algoritma yang sama dengan Subsoal 11 jika melakukan *compress* pada semua nilai tersebut. Kamu tidak perlu melakukan *compress* jika menggunakan *PBDS*.

Kompleksitas: $O(Q * log_2(N))$

D.13 Kode sumber

PBDS: https://ideone.com/9ABaAB
Binary Indexed Tree (BIT): https://ideone.com/ZhU1lP

E Hujan Megalopluvia

- penulis soal: Muhammad Faishol Amirul Mukminin, Galangkangin Gotera, Raden Fausta Anugrah Dianparama
- Tingkat kesulitan perkiraan: Menengah-Sulit
- Tag: *Deque, Linked List*

Definisikan segmen aktif ialah segmen kota dari interval $[L..R]$ yang sudah dialiri oleh hujan megalopluvia, terlepas dari sudah tenggelam atau belumnya kota tersebut. Perhatikan bahwa aliran airnya akan membentuk segmen tidak berbolong, dan segmen akan selalu mengalir ke kota yang lebih rendah, hingga bertemu kota yang dapat menjadi pembatas, dan membuat suatu kota yang lebih rendah tenggelam.

E.1 Subsoal 1

Perhatikan bahwa air akan terus mengalir ke tempat yang belum tenggelam. Dengan N yang kecil, dapat dilakukan simulasi secara manual layaknya ilustrasi.

Kompleksitas: $O(N^3)$

E.2 Subsoal 2

Mirip dengan subsoal 1, bedanya ialah terletak dari pencarian kota terendah pada suatu segmen, dapat dilakukan dengan data struktur minimum range query.

Kompleksitas: $O(N^2 \log N)$ **atau** $O(N^2)$

E.3 Subsoal 3

Perhatikan bahwa dari K, urutan tenggelam kota akan dimulai dari yang paling rendah, dan selalu mengikuti urutan dari 1 hingga N. Maka, urutan kota ke-i ada pada posisi ke-A_i

Kompleksitas: $O(N)$

E.4 Subsoal 4

Tanpa kehilangan sifat umum, sebenarnya sama saja dengan menyelesaikan permasalahan hujan selalu dimulai di kota pertama. Perhatikan bahwa saat dimulai di kota pertama, suatu kota tidak akan tenggelam apabila belum menemukan kota yang lebih tinggi darinya pada saat ini. Dapat disimpan stack yang menyimpan urutan kota secara menurun. Apabila suatu saat hujan mengaliri kota baru: Selama kota pada top stack masih lebih rendah dari kota baru, stack akan di-pop dan kota pada top stack akan tenggelam. Dapat dibuat sentinel Infinite pada kedua ujung kota, yang memastikan bahwa kota yang tertinggi akan

di-pop. Perhatikan bahwa kompleksitas waktu amortized O(1) untuk setiap kotanya.

Kompleksitas: $O(N)$

E.5 Subsoal 5

Mirip seperti subsoal 4, hanya saja kamu dapat menggunakan double-ended queue atau linked-list. Bedanya, kamu harus menyisakan setidaknya 2 kota. Untuk mengetahui air akan mengalir kemana, ke arah kiri atau kanan dari segmen saat ini. Implementasinya mirip dengan subsoal 4.

Kompleksitas: $O(N)$

E.6 Kode Sumber

https://ideone.com/BspuQi

F Angka Cantik

- penulis soal: Hocky Yudhiono
- Tingkat kesulitan perkiraan: Mudah - Menengah
- Tag: *Adhoc*

Pertama-tama, dijamin pasti ada jawaban. Tidak ada kemungkinan input yang menyebabkan jawaban -1.

F.1 Subsoal 1

Kamu bisa cek semua kemungkinan angka. Total hanya 200.000 angka yang mungkin di cek.

F.2 Subsoal 2

Buat angka T yang merupakan konkantenasi dari semua digit favorit bukan 0. Misal digit-digit tersebut $1, 4, 5, 6, 9$, $T = 14569$. Salah satu jawaban yang mungkin adalah $TTT0000$. Hal ini karena sifat dari angka yang habis dibagi 3 yaitu penjumlahan digit-digitnya harus habis dibagi 3.

Kompleksitas: $O(N)$

F.3 Subsoal 3

Kamu bisa membuat BFS bitmask sebagai berikut: misalkan $can[m][mask]$ adalah sebuah boolean yang menyatakan bisa/tidak membuat digit yang jika dimodulo M bernilai m dan telah menggunakan digit-digit pada $mask$. Pada mulanya $can[0][0] = 0$. Dari state (m, $mask$) kamu bisa transisi ke state $((m * 10)D_i, maskor2^i)$ dimana D_i adalah digit-digit cantik Bu Chanek.

Kompleksitas: $O(2^N M)$

F.4 Subsoal 4

Untuk kemudahan, anggap digit bukan nolnya T. Buat sekuens angka S =(, T, TT, TTT, $TTTT$, $TTTT..TTT$) dimana S_i adalah angka dengan i buat digit T. Hitung semua S_i modulo M, sebut m_i sebagai $S_i mod M$. Menghitung ini dengan cepat dapat dilakukan dengan modulo angka sebelumnya yaitu $m_{i+1} = (m_i * 10 + T) mod M$. Anggap ada i dan j dimana $m_i = m_j$. Maka didapati $S_i - S_j = 0 mod M$. Selain itu, angka ini juga masih menjaga sifat hanya terdiri dari digit-digit cantik Bu Chanek.

Kompleksitas: $O(M)$

F.5 Subsoal 5

Untuk setiap digit, kamu bisa mencari angka yang jika di mod = 0 dengan hanya menggunakan digit tersebut dan 0 (algoritma subsoal 4). Anggap angka-angka tersebut $S_1, S_2, S_3, ..., S_n$. Jelas bahwa jika $S_i mod M = 0, S_i * 10 mod M = 0$ jadi salah satu angka yang valid adalah $S_1 S_2 S_3 ... S_n$, konkatenasi dari setiap angka-angka.

Solusi alternatif adalah kamu membuat sebuah angka T yang merupakan konkatenasi dari semua digit-digit bukan 0. Contoh: jika digit favoritnya = 3, 4, 6, 7 maka $T = 3467$. Buat sekuens S dimana $S_i = T$ sebanyak i kali, i.e: $S_1 = 3456$, $S_2 = 34563456$, $S_3 = 345634563456$ dst. Hitung $M_1 = S_i mod M$. Pasti ada dua i, j, $i > j$ dimana $M_i = M_j$. Salah satu angka valid adalah $S_i - S_j$ karena berbentuk 3456345634560000..... Terlihat bahwa modulonya 0 dan memenuhi syarat-syarat yang diberikan.

Kompleksitas: $O(NM)$

F.6 Kode Sumber

https://ideone.com/rCAbe6

Senior Competitive Programming Contest

A Angka Cantik

- penulis soal: Hocky Yudhiono
- Tingkat kesulitan perkiraan: Mudah - Menengah
- Tag: *Adhoc*

Sudah di bahas di JCPC.

B Bintang Ojol

- penulis soal: Julian Fernando
- Tingkat kesulitan perkiraan: Mudah
- Tag: *Math*

Sudah di bahas di JCPC

C Cari Angin Keliling Singanesia

- penulis soal: Hocky Yudhiono
- Tingkat kesulitan perkiraan: Sulit
- Tag: *Min Cost Max Flow, Constructive Algorithm*

Permasalahan ini ialah Chinese Postman Problem pada graf directed. Jelas bahwa pada graf yang memiliki sirkuit euler, maka semua edge akan dikunjungi tepat sekali. Apabila tidak, maka akan ada edge yang akan dikunjungi lebih dari sekali.

kamu hanya perlu mengonstruksi directed graf baru yang memiliki sirkuit euler. Perhatikan bahwa pada graf yang memiliki sirkuit euler, jumlah derajat keluar dan masuk harus berjumlah sama. Dapat dilakukan Min Cost Maximum Flow pada graf yang kekurangan derajat masuk dan derajat keluar. Pada bagian source dihubungkan dengan semua verteks yang memiliki derajat keluar berlebih, sink dihubungkan dengan semua verteks yang memiliki derajat masuk berlebih, serta cost setiap pasang verteks ialah jarak terdekat antara dua verteks tersebut.

Perhatikan catatan tambahan bahwa graf baru hanya dapat dikonstruksi apabila graf terhubung secara kuat (strongly connected). Karena nilai maksimum N yang kecil, dapat dihitung jarang antara setiap pasang verteks menggunakan algoritma Floyd-Warshall. Sembari menghitung, dapat diketahui pula apakah graf tersebut terhubung secara kuat atau tidak. Floyd-Warshall juga akan sangat bermanfaat saat ingin mem-backtrack jawaban akhir.

C.1 Kode Sumber

https://ideone.com/BspuQi

D Distribusi Kelompok

- penulis soal: Hocky Yudhiono
- Tingkat kesulitan perkiraan: Mudah
- Tag: *Kombinatorika*

Anggap $P[i] = \sum_{j=1}^{i} A[i]$. Pertama asumsikan setiap kelompok memiliki jumlah anggota yang berbeda. Jawabannya pada kasus ini:

$$S = \Pi_{i=1}^{n} \binom{N - P[i-1]}{A[i]}$$

Namun bisa saja ada kelompok yang memiliki anggota yang sama seperti pada contoh masukan. Anggap ada K_i kelompok berjumlah i anggota. Sebuah konfigurasi dengan kelompok-kelompok ini dapat saling ditukar-tukar sehingga harus membagi S dengan $K!$. Lakukan pembagian ini untuk semua. Jadi, jawaban akhirnya:

$$\frac{\Pi_{i=1}^{n} \binom{N-P[i-1]}{A[i]}}{\Pi_{i=1}^{1000000} K_i!} \bmod 998244353$$

Untuk menghitung kombinasi dan faktorial dengan cepat, kamu bisa precompute semua faktorial dan inverse faktorial dalam $O(N)$ untuk faktorial dan $O(N \log N)$ untuk inverse faktorial.

Kompleksitas: $O(N)$

D.1 Kode Sumber

https://ideone.com/96JAN0

E Energi Kandang

- penulis soal: Galangkangin Gotera
- Tingkat kesulitan perkiraan: Sulit
- Tag: *Dynamic Programming, Deque*

Sudah di bahas di JCPC

F Fate of Chanek

- penulis soal: Galangkangin Gotera, Hocky Yudhiono
- Tingkat kesulitan perkiraan: Menengah
- Tag: Simulasi

Perhatikan nilai K yang kecil sehingga kamu bisa membuat algo yang mengiterasi semua kemungkinan string. Caranya sebagai berikut: untuk setiap indeks i, hitung $valid[i]$ yang menyatakan indeks pertama dimana substring $i..valid[i]$ mengandung string duplikat. Cara naifnya kemudian membuat daftar berisi indeks-indeks yang masih valid untuk suatu panjang L. Pada mulainya, semua indeks ada di daftar ini. Algoritmanya:

1. Iterasi setiap panjang L dari 1 - N
2. Untuk setiap panjang, hapus semua indeks yang sudah tidak valid.
3. Urutkan indeks-indeks yang tersisa secara lekiksografinya.

Jika melakukan langkah ketiga secara naif akan mendapatkan algoritma sorting $O(N^2 \log N)$ per panjang sehingga harus mencari cara mengurutkan lebih baik. Cara mengurutkannya adalah dengan memanfaatkan observasi bahwa saat mengurutkan indeks-indeks saat $L-1$, indeks sudah terurut dan terkumpulkan berdasarkan $L-1$ elemen pertama sehingga pada tahap ke-L untuk setiap kelompok indeks yang masih sama kamu hanya perlu membandingkan karakter terakhirnya saja. Hal ini dapat dilakukan dengan menyimpan pair $(first, second)$ untuk setiap indeks. $first$ menyatakan "kelompok" dari $L-1$ karakter pertama dan *second* menyatakan elemen pada posisi ke-L. Sorting kemudian dapat dilakukan pada pair ini dalam $O(N \log N)$.

Perhatikan bahwa kompleksitas *amortized* dari algoritma ini $O(K \log K)$, karena pada setiap langkah kamu hanya mengiterasi indeks-indeks yang masih valid sehingga total kamu akan menyentuk K buah string.

Kompleksitas $O(K \log K)$

1. Bonus 1: Kamu bisa mendapatkan solusi $O(K)$ jika mengganti sorting $O(N \log N)$ menjadi bucket sort
2. Bonus 2: Kamu bisa menyelesaikan permasalahan ini tanpa bergantung K dengan *suffix array* dan *segment tree* secara offline dalam $O((N + Q) \log N)$.

F.1 Kode Sumber

https://ideone.com/v7LrWz

G Gajah Malas

- penulis soal: Hocky Yudhiono
- Tingkat kesulitan perkiraan: Mudah-menengah
- Tag: *Adhoc*

  Anggap A sebagai panjang subsequence terpanjang dari array yang dimana $A_i = i$. Anggap B sebagai panjang subsequence terpanjang dari array dimana $B_i = N - i + 1$. Anggap A' sebagai panjang subsequence terpanjang dimana $A'_i = i$ pada array yang terbalik. Anggap B' sebagai panjang subsequence terpanjang dari array dimana $B'_i = N - i + 1$ pada array terbalik. Ada cara untuk mengurutkan array secara terurut menaik jika dan hanya jika $A + B \geq N$ atau $A' + B' \geq N$.

  Bukti jika: ambil semua elemen pada B sebagai operasinya lalu letakkan di belakang array. Array menjadi terurut menaik. **Bukti hanya jika**: Jika $A + B < N$, posisi elemen $A + 1$ pasti sebelum elemen A. Selain itu, posisi elemen $B - 1$ pasti terletak setelah posisi elemen B. Dalam pengambilan, ada 3 kemungkinan kasus:

  - pada subsequence yang diambil mengandung A tetapi tidak $A + 1$: jika diletakkan dibelakang maka akan terbentuk inversi sehingga harus kamu letakkan di depan. Namun jika kamu meletakkan di depan, kamu harus mengambil subsequence yang terurut terbalik dan berakhir di 1 (A'), tetapi kamu tahu bahwa $A' + B' < N$ sehingga mengambil ini akan menghasilkan inversi.
  - pada subsequence yang diambil mengandung $A + 1$ tetapi tidak A. Tidak mungkin diletakkan di depan karena akan menghasilkan inversi dengan A. Jika meletakkan di belakang maka harus mengambil B agar belakangnya valid. Namun kamu tahu $A + B < N$ sehingga pasti terdapat inversi.
  - jika mengambil A dan $A + 1$: tidak mungkin meletakkan di depan atau belakang (cukup jelas).

  Sehingga algoritmanya kamu bisa mencari A, B, A', B' dengan interasi sederhana lalu cek apakah ada yang melebihi N.

  Kompleksitas: $O(N)$

G.1 Kode sumber

https://ideone.com/JsO1DS

H Harta Karun Stompfec

- penulis soal: Dewangga Putra Sheradhien
- Tingkat kesulitan perkiraan: Medium
- Tag: *Hashing, Data structure*

Untuk setiap string kamu bisa menggunakan *Polinomial hash*:

$$P(S) = S_0 + S_1x + S_2x^2 + S_3x^3 + ... + S_{N-1}X^{N-1} mod P$$

Dengan struktur ini kamu bisa membangun struktur data segment tree untuk menghitung hash dari substring dengan mudah. Nyatakan A[i] sebagai apakah kata ke-i ada di kamus (0/1). kamu juga membuat segment tree berdasarkan informasi ini. Selanjutnya jawaban untuk suatu query yang dimulai di string ke-i dan berakhir di string ke-j adalah cek apakah substring parsial string ke-i dan j ada di kamus dan sum A[i] diantara i sampai j.

Disarankan untuk menghitung hash sebagai pair yang masing-masing di mod dengan dua prima berbeda seperti misalnya $10^9 + 7$ dan $10^9 + 9$. Beberapa cara hash yang bikin gagal:

1. Jika hanya hash dengan 1 prima, akan sangat besar peluang collision pada kasus dimana semua query tidak ada stringnya pada dictionary. kamu membandingkan 200.000 * 40.000 string sehingga peluang terjadi 1 collision mendekati 99.99%!
2. Jika tidak di mod dan menggunakan long long akan gagal pada kasus A B AB BA ABBA BAAB ABBABAAB BAABABBA... dimana string pada posisi ganjil memiliki peluang sangat besar collision dengan string pada posisi genap. Baca lebih lanjut di: https://codeforces.com/blog/entry/4898

Kompleksitas: $O(N \log N)$

H.1 Kode Sumber

: Segment tree: https://ideone.com/uf61Io
SQRT Decomposition: https://ideone.com/PU2R5l

I Illusory Tree

- penulis soal: Hocky Yudhiono
- Tingkat kesulitan perkiraan: Mudah
- Tag: *Tree checking*

kamu bisa membuat graph outputnya sebagai berikut: masukkan semua edge pada graph pertama ke sebuah set. Lalu saat iterasi graph kedua, jangan masukkan edge yang berada di set. Sisanya, semua edge ada di graph hasil. Kemudian sisa cek apakah tree. Pengecekan ini dapat dilakukan dengan cek apakah jumlah simpul pada graph ada sebanyak $N - 1$ dan graph sudah terhubung yang dapat di cek dengan DFS.

Kompleksitas: $O((K + L) \log N)$

I.1 Kode Sumber

https://ideone.com/rK3tZP