**Pembahasan AgriCode IPBPC –Week 3-**

1. Frog

Solusi : Greedy, Sorting

Kata kunci dari soal ini yaitu “seekor katak dapat diserang jika dan hanya jika Pi<=L” dan “diketahui pula ketika warga menyerang seekor katak maka akan mengurangi energi sebesar R setelah proses penyerangan” , karena pengurangan selalu tetap R maka akan semakin baik jika kita menyerang dari *Pi* yang paling besar terlebih dahulu (yang memungkinkan) karena nilai L akan terus berkurang sesudah menyerang *Pi* .untuk proses sorting nya, solusi yang diharapkan yaitu dengan menggunakan algoritma sorting yang O(N log N) (sort() pada STL c++ (quick sort), merge sort,heap sort dan sejenisnya)

1. Simulasi Prima

Solusi : Simulasi, Sieve of erathostenes, phi function

Simulasikan apa yang diperintahkan soal , untuk SieveNormal() dan EulerTotient() sudah terdapat dalam deskripsi hint nya, untuk NumDiv() kita hanya perlu memodifikasi sieve yang kita punya menjadi seperti ini :

for (int i=2; i<=100000; i++) {

for (int j=2; j<=100000/i; j++){

++f[i\*j];

}

}//sieve

kode tersebut akan menentukan apakah sebuah bilangan prima? (ditandai dengan f[i\*j]=0) dan berapa banyak factor yang dimiliki (ditandai dengan nilai f[i\*j]+2, +2 karena sieve diatas belum inclusive atau tidak memasukkan nilai 1 dan dirinya sendiri).

1. Index Tabel

Solusi : Deret Aritmatika (Sn)

Dari pola table dan pertanyaan yang diminta soal , sudah cukup jelas bahnya solusi yang diinginkan yaitu menghitung nilai *Sn* dari tiap baris, sebelumnya kita tuliskan terlebih dahulu rumus Sn :

\*Sn merupakan jumlah deret dari U1 hingga Un

\*Un merupakan suku ke-n

\*a merupakan suku pertama, b merupakan beda tiap suku.

Pada soal ini disyaratkan *ai>U****n,i-1*** maka masalah selanjutnya yang muncul yaitu bagaimana cara mencari *ai(suku pertama dari baris ke-i)* yang tepat

Setelah kita perhatikan maka untuk setiap baris ke-I didapatkan rumus

\*ceil merupakan fungsi pembulatan ke atas. +0.1 diperlukan karena kita ingin mencari nilai *ai* yang lebih dari (tidak sama dengan) *Un,i-1* , maka untuk kasus sama dengan, kita perlu menambahkan sedikit dengan angka <1 agar fungsi ceil bekerja.

Untuk nilai Un sudah cukup terlihat bahwa Un= kolom \* i, dan beda untuk tiap baris sama dengan i (b=i) untuk nilai n dari persamaan

Maka akan kita dapatkan

Maka hasil dari jumlah seluruhnya yang ditampung kedalam variable hasil untuk setiap baris (untuk setiap *i* ) yaitu :

Catatan :

hati-hati dalam proses modulo kita harus ingat bahwa

Teorama diatas berlaku juga untuk operasi pengurangan dan perkalian. Kemudian jangan lupa juga untuk menggunakan tipe data long long karena hasil bisa sangat besar , tetapi untuk setiap operasi perkalian dan penjumlahan pada soal ini dibuat tetap muat dalam long long (1<=N,M<=100000 , N\*M masih muat dalam long long) , Sehingga dalam mencari nilai Sn walaupun terdapat operasi pembagian , kita tidak memerlukan modulo invers ( karena masih muat di long long).

1. Ronde Nergi

Solusi : Sliding Window (geser segmen)

Inti dari soal ini yaitu kita harus mencari selisih dari nilai terbesar dan terkecil dari suatu segmen [left,right]. untuk cara *naïve* tentunya akan menghasilkan verdict TLE (tembus 20% testcase), lalu bagaimana cara mencarinya?(solusi full). Secara singkat algoritmanya seperti ini :

* Set 2 variabel left dan right dengan nilai 0,dua variabel ini akan kita jadikan sebagai index , set juga nilai min dan max sama dengan 0 lalu indexMax=0 dan indexMin=0.
* Selama *max-min<=k* lakukan geser variable right ke kanan(right+=1), dan untuk setiap nilai *data[right]* kita cek apakah *data[right]>max* atau *data[right]<min* , jika memenuhi diantara dua kondisi diatas maka update nilai max atau minnya (termasuk juga index dari nilai max/min kita simpan dalam variable indexMax/indexMin) .
* Selama proses penggeseran nilai right (right+1) maka compare segmen (right-left+1) dengan segmen maksimal sebelumnya(yang akan kita jadikan jawaban ) atau *segmen\_max=max(segmen\_max,right-left+1).*
* Ketika keluar dari *while (max-min<=k)* tentunya ada syarat yang tidak terpenuhi yaitu *max-min>k* atau *right>n(menandakan proses selesai, keluar dari loop while) ,* ketika kondisi *max-min>k ,* maka tentunya pergeseran nilai right membuat kondisi diatas tidak terpenuhi atau dengan kata lain membuat nilai min / max berubah, jika nilai min yang berubah , maka kita harus menggeser nilai left menjadi nilai indexMax+1 (begitupun dengan nilai max nya) , jika nilai max yang berubah maka kita harus menggeser nilai left menjadi indexMin+1 (begitupun nilai min nya) , lakukan hal ini hingga *max-min<=k* kembali terpenuhi.
* Jawaban dari soal ini ditampung dalam variable *segmen\_max*

1. Snow White II

Solusi : Binary Search the answer

Inti dari soal ini yaitu cari nilai X sedemikian sehingga jumlah semua nilai dari (sum(Hi-X),Hi>X ) > N. karena nilai M yang cukup besar yaitu 100000 , maka kita tidak dapat mencoba semua kemungkinan nilai X simulasikan terhadap M. karena nilai Hi yang mencapai 1000000000 , maka solusi naïve akan memberikan kompleksitas sebesar O(Hi\_Max\*M) yang tentunya TLE. Maka bagaimana cara mencarinya? , binary search sangat membantu didalam soal ini , karena kita tahu bahwa semakin kita menggeser nilai X ke-bawah maka selisih antara Hi-X akan semakin besar , maka untuk mendapatkan nilai dari sum(Hi-X)>=N , dengan hasil dari jumlah tersebut tidak berlebih atau hanya cukup memenuhi kondisi nilai sum(Hi-X)>=N, maka kita akan mendapatkan nilai X yang sebesar-besarnya , kondisi ini memenuhi sifat binary search, maka solusi dari soal ini yaitu lakukan binary search **terhadap jawaban(nilai X)** lakukan simulasi terhadap nilai X yang diperkirakan , jika Sum(Hi-X)>=N maka geser left ke kanan (left=left+1) sebaliknya geser right ke kiri (right=right-1).

1. Menyebarkan Permen

Solusi : Bruteforce

Soal ini cukup Obviuos , kita hanya tinggal melakukan if terhadap setiap jarak yang dikomparasi, hanya saja hati-hati dalam melakukan if , terdapat beberapa tricky case yang terlewatkan.

G.Jumlah makanan

Solusi : Kombinatorika , modulo multiplicative invers.

Soal ini menginginkan kita untuk mencari nilai dari

Untuk nilai N yang kecil , nilai kombinasi diatas masih dapat dihitung dengan mudah,(1<=N<=15) ,untuk nilai N yang besar tentunya kita akan mengalami kendala yang cukup besar , kendala-kendalanya sebagai berikut :

* Untuk setiap kombinasi C(N,K) kita harus simulasikan penghitungan kombinasi yang memerlukan waktu **(N-K) +K** (N\*(N-1)\*(N-2)\*(N-3)\*K) dan K!
* Factorial berkembang sangat besar sehingga akan sangat mudah suatu variable overflow.
* Karena dalam menghitung kombinasi C(N,K) terdapat operasi pembagian, dan sifat modulo untuk operasi pembagian berbeda dengan operasi lainnya (\*,+,-).

Dengan adanya kendala-kendala tersebut ada beberapa hal yang harus dilakukan :

* Karena sifat modulo dalam operasi pembagian tidak berlaku , maka hal yang harus dilakukan yaitu konversi operasi dari pembagian ke perkalian dengan *modulo multiplicative invers* :

<https://en.wikipedia.org/wiki/Modular_multiplicative_inverse>

dari link diatas , persamaan yang kita butuhkan yaitu :

Karena dari soal nilai m=1000000007 , yaitu bilangan prima, maka teorama diatas dapat diterapkan , atau dengan kata lain , untuk , maka dapat kita konversi menjadi

**)\*))**

**Ingat bahwa**

Setelah merumuskan rumus diatas , maka untuk mengitung *N!* dapat kita precompute di array agar tidak terjadi proses perhitungan yang berulang , kemudian dari persamaan diatas kembali didapatkan masalah baru , bagaimana menghitung  **dan** secara efisien?, jika kita melakukan looping biasa maka akan mendapatkan verdict TLE (karena 109+7 sangat besar), maka kita harus melakukan operasi perpangkatan tersebut dengan bantuan dari fast exponentiation (menggunakan konsep divide and conquer sehingga untuk melakukan proses pemangkatan hanya diperlukan waktu log2(n)).

H.Panen Sayuran

Solusi : Dynamic Programming , Exponentiation (Divide and Conquer), aturan perkalian

Dari syarat-syarat yang tertera, maka solusi yang pertama kali terfikirkan adalah kombinasi, tetapi dengan banyaknya syarat dan syaratnya bergantung dengan urutan sebelumnya, maka soal ini dapat diselesaikan dengan dynamic programming, dengan konstruksi fungsi rekursif.

Berdasarkan syarat-syarat berikut :

* Sayuran A harus dimasukkan pada X urutan pertama (X sayuran terbawah)
* Sayuran E harus dimasukkan pada Y urutan terakhir (Y sayuran teratas)
* Sayuran B harus dimasukkan setelah sayuran A atau B
* Sayuran C harus dimasukkan setelah sayuran D
* Sayuran D bebas, tidak ada syarat khusus untuk urutan pemasukan.
* Diharuskan terdapat minimal satu buah sayuran A pada tabung terbawah, dan satu sayuran E pada tabung teratas (pengecualian untuk M=1 , maka hanya sayuran A yang ditempatkan)

Maka kita dapat mengonstruksi fungsi rekursif sebagai berikut :

Fungsi diatas merupakan definisi untuk ***N=1 (1 tabung)* ,** dan jawaban yang diinginkan yaitu (karena pada tabung teratas harus jenis ‘E’). bagaimana dengan ***N>1*** tabung? Karena tabung-tabung identik maka banyaknya cara pada tabung 1 akan sama dengan banyaknya cara pada tabung 2, maka dari itu untuk N tabung banyaknya cara yaitu mod 109+7 (aturan perkalian). Untuk menghitung nilai  dapat menggunakan teknik top-down atau bottom-up , berikut contoh penerapan teknik bottom up untuk N=1 dan M=5 :

Tabel DP :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |
| **A** | 1 | 1 | 1 | 1 | 1 |
| **B** | 0 | 1 | 2 | 3 | 4 |
| **C** | 0 | 0 | 1 | 3 | 7 |
| **D** | 0 | 1 | 3 | 7 | 14 |
| **E** | 0 | 1 | 4 | 11 | **25** |