Editorial Final Schematics NPC Junior

17 Oktober 2021

Problem	Expected Difficulty	Author
Angka Sial	Easy	Naufal Faadhilah
Berhitung	Hard	Muhammad Amin
Crash Loyale	Medium-Hard	Muhamad Affan
Demam Shopaholic	Easy-Mediuum	Zydhan Linnar Putra
Eksplorasi Kota	Easy-Medium	Muhamad Affan
Fun Tour	Medium-Hard	Muhamad Affan
Good Game	Medium-Hard	Muhamad Affan
Hakuna Matata	Medium	Muhamad Affan
Indahnya Bintang di Langit	Medium	Muhamad Affan
Jaring-jaring Warna-warni	Medium	Vania Rizky J W

Tester

- **Andreas Cendranata**
- **Andreas Martin**
- **Fadhil Musaad**
- **Haniif Ahmad Jauhari**
- **Nurlita Dhuha Fatmawati**

Angka Sial

Pembuat Soal : Naufal Faadhilah

Tag : Math, Brute Force

Expected Difficulty : Easy

Batasan

Karena jumlah dari faktor bilangan N yang dicari bernilai 13, maka cukup dilakukan pencarian faktor bilangan N hingga faktor bernilai 13 saja. Kemudian, ketika telah didapat deretan faktor bilangan N, digunakan algoritma *brute-force* pada deret tersebut untuk mencari setiap kemungkinan sub-deret yang bernilai 13.

Kompleksitas: O(T * 2^{13})

B - Berhitung

Pembuat Soal : Muhammad Amin

Tag : Math, DP, Matrix Exponentiation

Expected Difficulty : Hard

Subsoal 1 (5 poin)

Perhatikan bahwa nilai a, b, c dan d hanya berentang dari -10 sampai 10. Kita dapat melakukan iterasi untuk semua nilai a, b, c dan d yang mungkin.

Kompleksitas akhir : $O(N)$.

Perhatikan disini nilai a,b,c,d dapat dipermutasikan, sehingga dapat diasumsikan bahwa a,b,c,d merupakan akar dari suatu polynomial sehingga:

$$f(x) = (x-a)(x-b)(x-c)(x-d)$$

$$f(x) = x^4 - \sum_{p=a}^{d} p\, x^3 + \sum_{p,q=a}^{d} pq\, x^2 - \sum_{p,q,r=a}^{d} pqr\, x + abcd$$

$$\sum_{p=a}^{d} p = a+b+c+d = i$$

$$\left(\sum_{p=a}^{d} p\right)^2 = \sum_{p=a}^{d} p^2 + 2\sum_{p,q=a}^{d} pq$$

$$\sum_{p,q=a}^{d} pq = \frac{i^2 - j}{2}$$

$$\sum_{p=a}^{d} p^3 - 3\left(\sum_{p,q,r=a}^{d} pqr\right) = \sum_{p=a}^{d} p\left(\sum_{p=a}^{d} p^2 - \sum_{p,q=a}^{d} pq\right)$$

$$\sum_{p,q,r=a}^{d} pqr = \frac{\sum_{p=a}^{d} p^3 - \sum_{p=a}^{d} p\left(\sum_{p=a}^{d} p^2 - \sum_{p,q=a}^{d} pq\right)}{3} = \frac{k - i(j - \frac{i^2-j}{2})}{3}$$

$$f(x) = x^4 - \sum_{p=a}^{d} p\, x^3 + \sum_{p,q=a}^{d} pq\, x^2 - \sum_{p,q,r=a}^{d} pqr\, x + abcd$$

$$f(x) = x^4 - i\,x^3 + \frac{i^2 - j}{2}\,x^2 - \frac{k - i\left(j - \frac{i^2 - j}{2}\right)}{3}\,x + abcd$$

$$f(a) + f(b) + f(c) + f(d) = 0$$

$$\sum_{p=a}^{d} p^4 - i \sum_{p=a}^{d} p^3 + \frac{i^2 - j}{2} \sum_{p=a}^{d} p^2 - \frac{k - i\left(j - \frac{i^2 - j}{2}\right)}{3} \sum_{p=a}^{d} p + 4abcd = 0$$

$$-4abcd = \sum_{p=a}^{d} p^4 - i \sum_{p=a}^{d} p^3 + \frac{i^2 - j}{2} \sum_{p=a}^{d} p^2 - \frac{k - i\left(j - \frac{i^2 - j}{2}\right)}{3} \sum_{p=a}^{d} p$$

$$abcd = \frac{l - i\,k + \frac{i^2 - j}{2}\,j - \frac{k - i\left(j - \frac{i^2 - j}{2}\right)}{3}\,i}{-4}$$

$$af(a) = a^5 - i\,a^4 + \frac{i^2 - j}{2}\,a^3 - \frac{k - i\left(j - \frac{i^2 - j}{2}\right)}{3}\,a^2 + \frac{l - i\,k + \frac{i^2 - j}{2}\,j - \frac{k - i\left(j - \frac{i^2 - j}{2}\right)}{3}\,i}{-4}\,a = 0$$

$$af(a) + bf(b) + cf(c) + df(d) = 0$$

n starting from 3 or 4

$$\sum_{p=a}^{d} p^{n+1} = i \sum_{p=a}^{d} p^n - \frac{i^2 - j}{2} \sum_{p=a}^{d} p^{n-1} + \frac{k - i\left(j - \frac{i^2 - j}{2}\right)}{3} \sum_{p=a}^{d} p^{n-2} - \frac{l - i\,k + \frac{i^2 - j}{2}\,j - \frac{k - i\left(j - \frac{i^2 - j}{2}\right)}{3}\,i}{-4} \sum_{p=a}^{d} p^{n-3}$$

$$Let \sum_{p=a}^{d} p^n = p_n$$

$$p_{n+1} = k_0 * p_n - k_1 * p_{n-1} + k_2 * p_{n-2} - k_3 * p_{n-3}$$

Kita dapat menghitung nilai p_n menggunakan *dynamic programming*.

Subsoal 2 (10 poin)

Pada subsoal ini, kita dapat melakukan perhitungan nilai DP untuk setiap *testcase*.

Kompleksitas akhir : $O(TN)$

Subsoal 3 (25 poin)

Pada subsoal ini, melakukan perhitungan di tiap *testcase* akan menyebabkan TLE. Maka dari itu,kita butuh melakukan prekomputasi sebelum menjawab *testcase*.

Kompleksitas akhir : $O(N)$

Subsoal 4 (60 poin)

Dikarenakan *constraint* yang besar, kita tidak dapat melakukan prekomputasi untuk semua kemungkinan nilai N.

Karena terjadi linear occurrence, maka kita dapat menggunakan Matrix Exponentiation yang membuat time complexity perhitungan untuk setiap test case menjadi O(log(N)) dengan matrixnya adalah

$$\begin{matrix} k_0 & k_1 & k_2 & k_3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{matrix}$$

Kompleksitas: O(T * log(N))

Crash Loyale

Pembuat Soal : Muhamad Affan

Tag : Dynamic Programming, Probability

Expected Difficulty : Medium-Hard

Subsoal 1 (8 poin)

Untuk kasus $N = 1$, kartu yang mungkin terambil hanyalah kartu-kartu yang berada pada dek tangan awal. Sehingga, nilai harapan dari total *elixir* yang akan terpakai bisa didapatkan dari penjumlahan nilai *elixir* pada kartu awal yang dibagi dengan 4.

Total kompleksitas: $O(1)$

Subsoal 2 (18 poin)

Sama seperti subsoal 1, hanya saja karena kasus ini total elixir yang terpakai belum tentu habis dibagi 4. Dengan menggunakan *fermat little theorem* kita bisa menghitung nilai dari $\frac{totalelixir}{4} \ mod \ 10^9 + 7$ menjadi $totalelixir * 4^{10^9+5}$ yang dapat dihitung menggunakan *modular exponentiation*.

Total kompleksitas: $O(\log MOD)$.

Subsoal 3 (25 poin)

Secara total, paling banyak ada 8^5 kemungkinan urutan yang mungkin. Karena nilai tersebut sangat kecil, kita dapat mengecek untuk setiap kemungkinan apakah kemungkinan tersebut valid atau tidak. Jika valid, maka kita akan menghitung jumlah *elixir* yang akan terpakai untuk kemungkinan tersebut.

Total kompleksitas : $O(8^N \times N)$

Subsoal 4 (13 poin)

Perhatikan bahwa pada kasus ini, untuk setiap ronde, jumlah *elixir* yang digunakan pasti sama. Maka dari itu, total *elixir* yang digunakan pasti ada sebanyak $A_i \times N$.

Subsoal 5 (36 poin)

Pertama-tama, kita dapat merepresentasikan sebuah dek tangan sebagai sebuah *bitmask* yang terdiri dari 8 bit dengan bit ke-*i* bernilai 1 jika kartu ke-*i* sedang berada di dek tangan, dan 0 jika sebaliknya.

Kita dapat mendefinisikan $DP[i][j]$ = sebagai nilai harapan pemakaian *elixir* sampai ronde ke-*i,* serta dek tangan setelah ronde ke-*i* bernilai *j.*

Serta $prob[i][j]$ adalah probabilitas kita memiliki dek tangan bernilai *j* setelah ronde ke-i.

Didefinisikan pula sebuah *bitmask* x sebagai *substitute* dari sebuah *bitmask* y jika bit yang menyala pada hasil $x\ xor\ y$ berjumlah **tepat** 2 bit. Sebagai contoh, jika $x = (10110010)_2$ dan $y = (10011010)_2$ maka x merupakan *substitute* dari y.

Nilai $prob[i][j]$ dapat dihitung menggunakan rumus :

$$prob[i][j] = \sum_{j'} \frac{1}{16} \times prob[i-1][j']$$

Dimana j' merupakan *substitute* dari *j*.

Ini bisa didapatkan dari probabilitas nilai j' ber-transisi menjadi j dalam 1 ronde adalah $\frac{1}{16}$.

Dari sini, kita juga dapat melihat bahwa saat bertransisi dari *mask* j' menjadi j, terdapat 1 buah bit yang menyala pada j' dan mati pada j. Misalkan bit tersebut merupakan bit ke-k, maka artinya pada saat ronde ke-i, kita menggunakan kartu dengan indeks k.

Dengan begitu, kita dapat merumuskan $DP[i][j]$ sebagai :

$$DP[i][j] = \sum_{j'} \frac{1}{16} \times (A_k \times prob[i-1][j'] + DP[i-1][j']).$$

Dimana j' merupakan *substitute* dari *j*.

Jawaban bisa didapatkan dari $\sum_{mask} DP[N][mask]$.

Perhatikan bahwa jumlah *bitmask* yang valid hanya ada sebanyak $\binom{8}{4} = 70$ sehingga total transisi untuk tiap ronde ada sebanyak 70×4^2

Total kompleksitas : $O(N \times 70 \times 4^2)$

Demam Shopaholic

Pembuat Soal : Zydhan Linnar Putra

Tag : Meet in the Middle,Binary Search

Expected Difficulty : Easy-Medium

Subsoal 1 (10 poin)

Dapat dilakukan brute-force bitmask untuk setiap barang yang ada dengan mengecualikan kombinasi barang dengan harga 0 (Tidak membeli apapun)

Kompleksitas: $O(2^n)$

Subsoal 2 (90 poin)

Brute force dapat dilakukan dengan dua array $left$ dan $right$ lalu iterasi setiap kombinasi pada array left kemudian diikuti dengan mencari kombinasi pada array

$right$ dengan jumlah yang kurang dari atau sama dengan C.

Kompleksitas: $O(2^{\frac{N}{2}} \log 2^{\frac{N}{2}})$

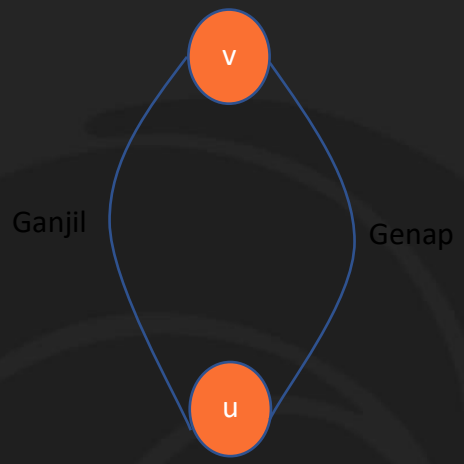
Eksplorasi Kota

Pembuat Soal : Muhamad Affan

Tag : Graph

Expected Difficulty : Easy-medium

Observasi : jika terdapat 2 buah *path* yang tidak saling berpotongan dari U ke V dengan paritas panjang yang berbeda, maka *path* tersebut akan membentuk sebuah *cycle* dengan banyak verteks yang berjumlah ganjil.



Maka dari itu, soal ini dapat direduksi menjadi mengecek apakah terdapat sebuah *cycle* ganjil atau tidak.

Subsoal 1 (12 poin)

Kita dapat melakukan DFS untuk setiap verteks dan mengecek keberadaan *cycle* ganjil.

Kompleksitas Akhir : $O(N \times M)$

Subsoal 2 (23 poin)

Graf terdiri dari 1 buah *connected component* dan berbentuk sebagai sebuah *tree* dengan tepat 1 buah *cycle.* Kita dapat membuat sebuah *spanning tree* dari graf tersebut terlebih dahulu. Dengan begitu , *edge* yang tidak termasuk pada *spanning tree* tersebut merupakan *edge* yang membuat *cycle*.

Misalkan *edge* tersebut menghubungkan verteks U dan V, maka besar dari *cycle* yang terbentuk adalah $depth_u + depth_v - 2 \times depth_{lca} + 1$. Dimana LCA merupakan *lowest common ancestor* dari U dan V.

Kompleksitas Akhir : $O(N + M)$

Subsoal 3 (23 poin)

Graf terdiri dari beberapa buah *connected component* dimana setiap *connected component* berbentuk sebuah *cycle*. Untuk itu, kita dapat melakukan *disjoint set union* ataupun dfs untuk mengetahui ukuran masing-masing *connected component.*

Kompleksitas akhir : $O(N + M)$

Subsoal 4 (42 poin)

Observasi : sebuah graf memiliki *cycle* ganjil jika dan hanya jika graf tersebut bukan merupakan *bipartite*.

Dengan begitu, kita hanya perlu mengecek jika terdapat *connected component* yang bukan merupakan *bipartite*.

Kompleksitas akhir : $O(N + M)$

Fun Tour

Pembuat Soal : Muhamad Affan

Tag : Graph,Adhoc

Expected Difficulty : Medium-hard

WLOG kita asumsikan $N \leq M$, serta $c_1 \leq c_2$.

Subsoal 1 (9 poin)

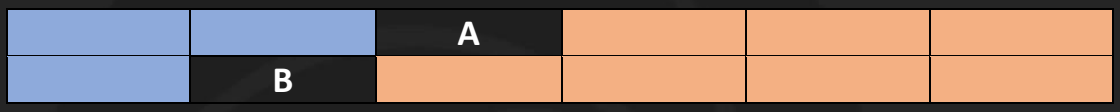
Karena grid hanya terdiri dari 1 baris, maka hanya terdapat tepat 1 jalan yang menghubungkan tiap pasang kota dengan panjang $c_2 - c_1 + 1$.

Kompleksitas akhir : $O(1)$

Subsoal 2 (23 poin)

Terdapat 2 kondisi berbeda untuk kasus ini

- $c_1 = c_2$ dan $c_1 \neq 1, c_1 \neq M$ atau $c_1 = c_2 - 1$.



  Pada kondisi ini, jika kita menghapus kedua titik, maka grid akan terbagi menjadi 2 bagian dimana sebuah *path* yang menghubungkan titik A dan B pasti melewati salah satu dari 2 bagian tersebut. Sehingga panjang *path* terpanjang adalah ukuran maksimal dari kedua bagian yang terbentuk. Atau dengan kata lain $\max(c_1 + c_2, 2M - c_1 - c_2 + 2)$.

- Selain kasus di atas.
  Misalkan kita mewarnai sebuah titik $X(r, c)$ dengan warna putih jika $r + c$ bernilai genap dan hitam jika sebaliknya. Dapat dilihat bahwa jika titik A dan B memiliki warna yang berbeda, maka kita dapat membuat sebuah *path* dengan panjang $N * M$. Jika A dan B memiliki warna yang sama, maka kita dapat membuat sebuah *path* dengan panjang $N * M - 1$.

Kompleksitas akhir : $O(1)$

Subsoal 3 (10 poin)

Karena hanya terdapat paling banyak 10 buah titik, maka kita dapat mencoba semua kemungkinan urutan *path* yang akan dikunjungi.

Kompleksitas akhir : $O((NM)!)$

Subsoal 4 (47 poin)

Seperti pada subsoal 2, kita dapat mewarnai sebuah titik $X(r, c)$ dengan warna putih jika $r + c$ bernilai genap dan hitam jika sebaliknya. Dapat diperhatikan bahwa dengan ini, grid dapat dimodelkan sebagai sebuah *bipartite graph*. Perhatikan bahwa dalam *bipartite graph*, setiap *path* akan terdiri dari verteks hitam dan putih yang saling bergantian.

Dengan begitu, kita dapat membagi permasalahan menjadi beberapa kasus :

1. Grid berukuran genap, serta A dan B memiliki warna yang berbeda.
   Perhatikan bahwa pada *bipartite graph*, sebuah *path* yang memiliki panjang genap pasti berawal dan berakhir di dua warna yang berbeda. Karena jumlah verteks berjumlah genap, sebuah *path* yang mengunjungi semua verteks pada grid pasti memiliki panjang genap serta memiliki warna awal dan akhir yang berbeda. Dengan begitu, jawaban untuk kasus ini adalah $N * M$.

2. Grid berukuran ganjil, serta A dan B memiliki warna putih.
   Seperti observasi pada kasus sebelumnya, sebuah *path* yang memiliki panjang ganjil pasti berawal dan berakhir di dua warna yang sama. Karena pada grid berukuran ganjil banyaknya verteks putih berjumlah lebih besar 1 dari banyaknya verteks hitam, maka sebuah *path* yang mengunjungi semua verteks pada grid pasti berawal dan berakhir pada verteks berwarna putih. Dengan begitu, jawaban untuk kasus ini adalah $N * M$.

3. Grid berukuran genap, serta A dan B memiliki warna yang sama.
   Sama seperti observasi pada kasus 1, karena A dan B memiliki warna yang sama, maka *path* yang menghubungkan harus memiliki panjang ganjil. Dengan begitu, panjang *path* maksimal yang bisa dibentuk adalah $N * M - 1$.

4. Grid berukuran ganjil, serta A dan B memiliki warna yang berbeda.
   Kasus ini dapat diselesaikan dengan analisis yang sama dengan kasus 3.

5. Grid berukuran ganjil, serta A dan B memiliki warna hitam.
   Kasus ini dapat diselesaikan seperti kasus 2, tetapi pada kasus ini jika panjang *path* sebesar $N * M$, maka *path* tersebut pasti hanya berawal serta berakhir pada verteks putih. Maka dari itu, jika kedua verteks berwarna hitam, maka panjang *path* terpanjang yang mungkin adalah $N * M - 2$.

6. Grid memenuhi syarat berikut:
   - Banyaknya baris berjumlah 3.
   - Banyaknya kolom berjumlah genap.
   - A berwarna hitam serta B berwarna putih.
   - $r_1 = 2$ dan $c_1 < c_2$ atau $r_1 \neq 2$ dan $c_1 < c_2 - 1$

   Contoh:

A			
	B		

   Meskipun kasus ini termasuk pada kasus 1, dapat dibuktikan bahwa kita tidak dapat membuat $path$ sepanjang $N * M$ dari A ke B. Pada kasus ini, panjang *path* terpanjang yang mungkin adalah $N * M - 2$.

Good Game

Pembuat Soal : Muhamad Affan

Tag : Adhoc

Expected Difficulty : Hard

Subsoal 1 (7 poin)

Observasi : hanya memilih 1 buah label akan selalu menyebabkan Elsi kalah. Karena itu, pilihan kita hanyalah memilih 0 atau 2 buah label kartu.

Jika Elsi memilih kedua label kartu, maka sudah pasti Elsi akan menang.

Jika Elsi tidak memilih label kartu sama sekali, kita dapat mensimulasikan permainan yang dilakukan dengan menggunakan *dynamic programming* untuk mengecek apakah Elsi akan menang atau tidak.

Kompleksitas akhir : $O(N^3)$

Subsoal 2 (15 poin)

Observasi : Perhatikan bahwa jika $freq[i]$ adalah jumlah kemunculan nilai i pada array A, maka kita dapat mereduksi permainan menjadi sebuah *nim game* klasik pada array $freq$. Dengan begitu, Elsi akan menang jika hasil xor dari array $freq$ bernilai 0.

Dengan menggunakan observasi pada subsoal 1, kita hanya perlu mengecek jika Elsi dapat menang jika ia tidak memilih label kartu sama sekali. Ini dapat kita lakukan dengan mengecek jika hasil xor dari frekuensi kedua label bernilai 0. Atau dengan kata lain, frekuensi kemunculan kedua label sama.

Kompleksitas akhir : $O(1)$

Subsoal 3 (7 poin)

Dengan menggunakan observasi yang sama dari subsoal 2, kita dapat menyelesaikan subsoal ini dengan mengecek untuk semua kemungkinan *subset* yang bisa dipilih jika hasil xor frekuensi dari elemen *subset* tersebut bernilai 0.

Kompleksitas akhir : $O(2^N * N)$

Subsoal 4 (14 poin)

Karena $A_i = i$, maka $freq[i] = 1$ untuk $1 \leq i \leq N$. Karena itu, *subset* yang memiliki hasil xor bernilai 0 hanyalah $subset$ yang berukuran genap. Jumlah *subset* yang berukuran genap ada sebanyak 2^{N-1}.

Kompleksitas akhir : $O(N)$

Subsoal 5 (26 poin)

Untuk menghitung jumlah *subset* yang memiliki hasil xor bernilai 0, kita dapat menggunakan *dynamic programming*. Misalkan kita definisikan $DP[i][j]$ adalah banyak cara membuat *subset* dari i bilangan pertama dengan hasil xor bernilai j.

Kita dapat menghitung nilai $DP[i][j]$ dengan cara :

$$DP[i][j] = DP[i-1][j] + DP[i-1][j\ xor\ freq[i]]$$

Untuk mendapatkan hasil xor bernilai j, kita dapat mengambil $subset$ dari $i-1$ yang memiliki hasil xor bernilai j, atau kita dapat mengambil *subset* dari $i-1$ yang memiliki hasil xor bernilai $j\ xor\ freq[i]$ lalu memasukan i ke *subset* tersebut.

Jawaban bisa didapatkan dari $DP[N][0]$.

Kompleksitas akhir : $O(N^2)$

Subsoal 6 (31 poin)

Karena batasan yang besar, kita tidak dapat menggunakan solusi *dynamic programming* pada subsoal 5. Tetapi, kita dapat menggunakan ide dari solusi tersebut.

Lemma : $DP[i][j]$ bernilai 0 jika j tidak dapat dibentuk dari hasil xor *subset-subset* yang diambil dari i bilangan pertama. Serta $DP[i][j]$ bernilai sama untuk semua j dimana j bisa dibentuk dari hasil $xor\ subset$ dari i bilangan pertama.

Bukti untuk bagian pertama lemma cukup *trivial*.

Kita dapat membuktikan bagian kedua dari *lemma* menggunakan induksi.

Misalkan S_i adalah nilai-nilai hasil xor yang mungkin terbentuk dari $subset\ i$ bilangan pertama. Dengan $S_i = \{0\}$.

Dapat dilihat bahwa pada saat $i = 0$, $DP[0][0]$ bernilai 1 dan $DP[0][j]$ bernilai 0 untuk $1 \leq j \leq N$.

Pada saat melakukan transisi di indeks i, hanya terdapat 2 kemungkinan, yaitu $Freq[i]$ bukan merupakan elemen dari S_{i-1}, atau sebaliknya. Kita dapat meninjau masing-masing dari kedua kemungkinan ini.

Kemungkinan 1 : $freq[i]$ bukan merupakan elemen dari S_{i-1}

Saat melakukan transisi pada indeks i, untuk setiap nilai j, hanya paling banyak 1 diantara nilai $DP[i-1][j]$ dan $DP[i-1][j\ xor\ freq[i]]$ yang bernilai tidak sama dengan 0. Karena hanya paling banyak 1 diantara j dan $j\ xor\ freq[i]$ yang merupakan elemen dari S_{i-1}. Ini dikarenakan jika j dan $j\ xor\ freq[i]$ keduanya merupakan elemen dari S_{i-1}, maka $freq[i]$ pasti juga merupakan elemen dari S_{i-1} yang menimbulkan kontradiksi dengan kondisi awal.

Misalkan $DP[i-1][j] = K$ untuk setiap j merupakan elemen dari S_{i-1}. Maka, $DP[i][j] = DP[i-1][j] + DP[i-1][j\ xor\ freq[i]] = K$ untuk setiap j merupakan elemen dari S_i dan $DP[i][j] = 0$ jika sebaliknya.

Kemungkinan 2 : $freq[i]$ merupakan elemen dari S_{i-1}

Mirip seperti observasi pada kemungkinan pertama,pada kasus ini, saat melakukan transisi pada indeks i, kedua nilai $DP[i-1][j]$ dan $DP[i-1][j\ xor\ freq[i]]$ bernilai tak 0 jika j merupakan elemen dari S_{i-1}.

Misalkan $DP[i-1][j] = K$ untuk setiap j merupakan elemen dari S_{i-1}. Maka, $DP[i][j] = DP[i-1][j] + DP[i-1][j\ xor\ freq[i]] = 2K$ untuk setiap j merupakan elemen dari S_i dan $DP[i][j] = 0$ jika sebaliknya.

Dengan begitu, kita dapat melihat bahwa pada saat melakukan transisi pada indeks i, jika $freq[i]$ tidak bisa dibentuk dari hasil $xor\ subset$ dari *i* indeks pertama, maka nilai-nilai DP tidak akan berubah. Dan jika sebaliknya, maka nilai DP akan bertambah menjadi 2 kali lipat.

Dengan begitu, untuk menghitung jawaban, kita hanya perlu meng-*keeptrack* hasil hasil *xor subset* pada tiap indeks di dalam sebuah set. Jika nilai frekuensi indeks tersebut ada di dalam set, maka jawaban kita kalikan dengan 2. Jika tidak, maka kita lakukan *update* pada isi set tersebut dengan memasukkan hasil xor elemen-elemen pada set dengan $freq[i]$.

Perhatikan bahwa jumlah elemen pada set tidak akan melebihi 2^k dimana k merupakan bilangan terkecil yang memenuhi $2^k > N$.

Kompleksitas akhir : $O(N \log(Max(A_i)))$

Hakuna Matata

Pembuat Soal : Muhamad Affan

Tag :Dynamic Programming

Expected Difficulty : Medium

Subsoal 1 (9 poin)

Untuk subsoal ini, kita dapat mencoba semua kemungkinan formasi barisan.

Kompleksitas akhir : $O(N! * N)$

Subsoal 2 (15 poin)

Kita dapat membagi urutan bilangan menjadi berbentuk seperti ini:

$$\{A\} < x > y < \{B\}$$

Dengan A dan B berbentuk sebuah barisan *increasing*.

Perhatikan bahwa bilangan-bilangan yang bernilai kurang dari x tidak mungkin berada pada bagian B. Begitu juga bilangan-bilangan yang bernilai lebih dari y tidak mungkin berada pada bagian A.

Sedangkan bilangan-bilangan yang bernilai diantara x dan y dapat ditempatkan di A maupun B.

Sehingga, untuk suatu pasang x dan y, terdapat 2^k barisan yang memenuhi aturan. Dimana k merupakan banyaknya bilangan yang bernilai diantara x dan y.

Subsoal 3 (66 poin)

Observasi : bilangan-bilangan yang ada tidak penting, yang penting hanya urutan bilangan tersebut dari paling rendah ke paling tinggi.

Kita dapat memandang permasalahan seperti memasukkan bilangan-bilangan tersebut satu persatu ke sebuah *array* dari bilangan terkecil hingga bilangan terbesar.

Misalkan kita sudah memasukkan i bilangan pertama serta terdapat j buah indeks dimana $A_j > A_{j+1}$. Misalkan indeks-indeks ini disebut sebagai indeks turun.

Saat memasukkan bilangan ke-$i + 1$, jika kita memasukkan bilangan tersebut di indeks sebelum indeks-indeks turun, maka jumlah indeks turun akan bertambah sebanyak 1. Sebaliknya, jika kita memasukkan bilangan tersebut di indeks setelah indeks-indeks turun, maka jumlah indeks turun akan tetap.

Dengan begitu, kita dapat mendefinisikan $DP[i][j]$ sebagai jumlah konfigurasi menggunakan i bilangan terkecil dengan indeks turun sebanyak j.

Kita dapat menghitung $DP[i][j]$ dengan cara

$$DP[i][j] = (j+1) * DP[i-1][j] + (i-j) * DP[i-1][j-1]$$

Jawaban bisa didapatkan dari nilai $DP[N][K]$.

Kompleksitas akhir : $O(N^2)$

Subsoal 4 (10 poin)

Kita dapat melakukan hal yang sama seperti pada subsoal 3, tetapi penghitungan DP harus menggunakan teknik *flying table* agar tidak melewati *memory limit*.

Kompleksitas akhir : $O(N^2)$

Kompleksitas memori : $O(N)$

Indahnya Bintang di Langit

Pembuat Soal : Muhamad Affan

Tag : Geometry,Two Pointer

Expected Difficulty : Medium

Subsoal 1 (18 poin)

Perhatikan bahwa ketiga titik membentuk sebuah segitiga. Dengan begitu, kita bisa menggunakan aturan *cosinus* untuk mencari sudut-sudut pada segitiga tersebut.

Terdapa *tricky case* dimana segitiga yang diberikan mempunyai luas 0 atau dengan kata lain ketiga titik berada pada 1 garis.

Kompleksitas akhir : $O(1)$

Subsoal 3 (26 poin)

Perhatikan bahwa karena semua titik berada pada garis yang sama, maka sudut yang mungkin terbentuk hanyalah $0°$ ataupun $180°$. Untuk *triplet* (A, B, C) yang berpusat di titik A, triplet tersebut akan membentuk sudut $0°$ jika $B_y, C_y \leq A_y$ atau $B_y, C_y \geq A_y$.

Misalkan L_i adalah jumlah titik yang memiliki koordinat y lebih kecil daripada titik i dan R_i adalah jumlah titik yang memiliki koordinat y lebih besar daripada titik i. Maka jawaban bisa diperoleh dari

$$\sum_{i=1}^{N} \binom{L_i}{2} + \binom{R_i}{2}$$

Kompleksitas akhir : $O(N)$

Subsoal 2 (16 poin)

Kita dapat mengiterasi semua kemungkinan triplet (A, B, C) lalu mengecek besar sudut yang terbentuk dari triplet tersebut.

Kompleksitas akhir : $O(N^3)$

Subsoal 4 (40 poin)

Untuk suatu titik A yang menjadi titik pusat sudut, kita dapat melakukan *sorting* pada titik-titik lain berdasarkan *polar angle* terhadap titik A. Untuk setiap indeks L, kita dapat mencari indeks terbesar R dimana sudut yang terbentuk dari triplet (A, L, R) memiliki ukuran lebih kceil dari $90°$. Ini dapat dilakukan menggunakan teknik *two pointer* dimana untuk indeks pertama, kita dapat melakukan pencarian secara naif hingga menemukan R terbesar dimana triplet (A, L, R) membentuk sudut yang lebih kecil dari $90°$.

Setelah itu, kita dapat menggeser pointer L ke titik selanjutnya dan kembali menggeser pointer R hingga sudut yang terbentuk sudah lebih dari 90 °.

Dengan begitu, kita dapat menghitung jumlah triplet dengan titik pusat A dalam waktu $O(N)$.

Kita dapat menghitung jawaban akhir dengan mengulang langkah diatas untuk semua N titik yang ada.

Kompleksitas akhir : $O(N^2)$

Jaring Warna-Warni

Pembuat Soal : Vania Rizky J W

Tag : Math

Expected Difficulty : Medium

Untuk semua subsoal berlaku

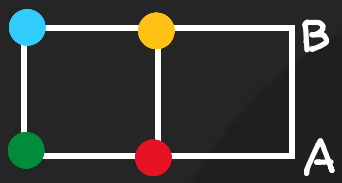
$1 \leq M, N \leq 10^{18}$

Subsoal 1 (7 poin)

$1 \leq M \times N \leq 4$

Untuk $M = N = 1$ sudah jelas ada 24 pola pewarnaan

Untuk $(M, N) = (1, 2)$ atau $(2, 1)$



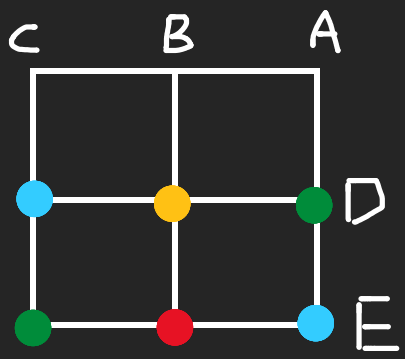
Ada 24 cara untuk memberi warna pada grid 1×1

Untuk membuat grid 1×2 maka ditambahkan titik A dan B sesuai pada gambar. Terdapat 2 pilihan warna untuk titik A. Ketika titik A sudah ditentukan warnanya, titik B hanya mempunyai 1 pilihan.

Maka banyaknya pola pewarnaan adalah $24 \times 2 = 48$

Untuk $M = N = 2$ melanjutkan pewarnaan dari grid 1×2 atau 2×1, ada 2 pola

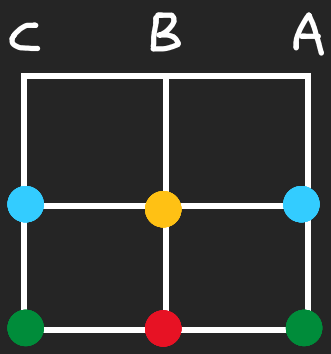
Pola 1



Pada pola 1, titik A hanya boleh memiliki 1 warna yaitu biru. Karena jika A merah, maka B biru. Namun di persegi yang memuat titik B dan C sudah memuat biru.

Maka untuk pola 1 ini hanya ada 1 cara pewarnaan dari grid 1×1 yang titik D nya bewarna hijau dan E bewarna biru. Jadi ada $24 \times 1 = 24$ pola pewarnaan

Pola 2



Pada pola 2, titik A mempunyai 2 pilihan warna yaitu merah dan hijau

Maka untuk pola 2 ini ada 2 cara pewarnaan dari grid 1×1 yang titik D nya bewarna hijau dan E bewarna biru. Jadi ada $24 \times 2 = 48$ pola pewarnaan

Jadi banyak pewarnaan untuk grid 2×2 adalah $24 + 48 = 72$ pewarnaan

Observasi yang sama dilakukan untuk

$(M, N) = (1, 4)$ atau $(4,1)$ ada 192 pewarnaan

$(M, N) = (1, 3)$ atau $(3,1)$ ada 96 pewarnaan

Kompleksitas : O(1)

Subsoal 2 (38 poin)

$1 \leq M, N \leq 10^3$

Pada grid $M \times N$ terdapat $M + 1$ baris horizontal dan $N + 1$ garis vertikal

Jika baris M menggunakan 3 atau 4 warna, maka terdapat keadaan dimana

$$M : \ldots abc \ldots$$

maka untuk baris ke $M + 1$ hanya ada 1 pewarnaan yaitu

$$M : \ldots abc \ldots$$

$$M + 1 : \ldots cda \ldots$$

Perhatikan bahwa untuk baris $M-1$ ternyata hanya ada 1 pewarnaan pula, yaitu seperti pada pewarnaan baris $M+1$

Maka jika grid $M \times N$ dengan baris paling akhirnya menggunakan 3 atau 4 warna maka ketika akan diekspansi menjadi grid $(M+1) \times N$, banyak pewarnaan pada kolom N tidak akan terpengaruh dengan penambahan baris ke $M+1$

Jika baris M menggunakan 2 warna maka ada 2 kemungkinan pewarnaan untuk baris $M+1$

Pola 1

M : ... $ababab$ ...

$M+1$: ... $cdcdcd$ ...

Pola 2

M : ... $ababab$ ...

$M+1$: ... $dcdcdc$ ...

Maka jika grid $M \times N$ dengan baris paling akhirnya menggunakan 2 warna maka ketika akan diekspansi menjadi grid $(M+1) \times N$, banyak pewarnaan pada kolom N terpengaruh dengan penambahan baris ke $M+1$

Perhatikan bahwa jika sebuah baris menggunakan 2 warna, maka keseluruhan baris akan menggunakan 2 warna. Jika sebuah baris menggunakan 3 atau 4 warna, maka keseluruhan baris akan menggunakan 3 atau 4 warna. Keadaan yang sama diterapkan untuk kolom.

Akan dibuat sistem rekurensinya

Misalkan

$Q_{M,N}$ = banyak pewarnaan jika baris M menggunakan 2 warna dan kolom N 2 warna

$R_{M,N}$ = banyak pewarnaan jika baris M menggunakan 2 warna dan kolom N 3 warna

$S_{M,N}$ = banyak pewarnaan jika baris M menggunakan 3 warna dan kolom N 2 warna

$T_{M,N}$ = banyak pewarnaan jika baris M menggunakan 3 warna dan kolom N 3 warna

Maka

$$\begin{bmatrix} Q_{1,1} & R_{1,1} \\ S_{1,1} & T_{1,1} \end{bmatrix} = 24 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} Q_{M,N} & R_{M,N} \\ S_{M,N} & T_{M,N} \end{bmatrix} = \begin{bmatrix} Q_{M-1,N} & Q_{M-1,N} + 2R_{M,N} \\ S_{M-1,N} & T_{M-1,N} \end{bmatrix} = \begin{bmatrix} Q_{M,N-1} & R_{M,N-1} \\ Q_{M,N-1} + 2S_{M,N-1} & T_{M,N-1} \end{bmatrix}$$

Berdasarkan analisa konsep, perhatikan bahwa $T_{m,n} = 0$ untuk seluruh M, N

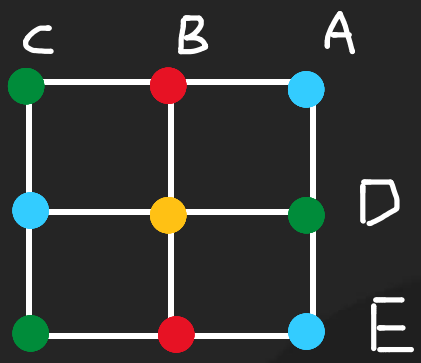
Kompleksitas : $O(N)$

Subsoal 3 (43 poin)

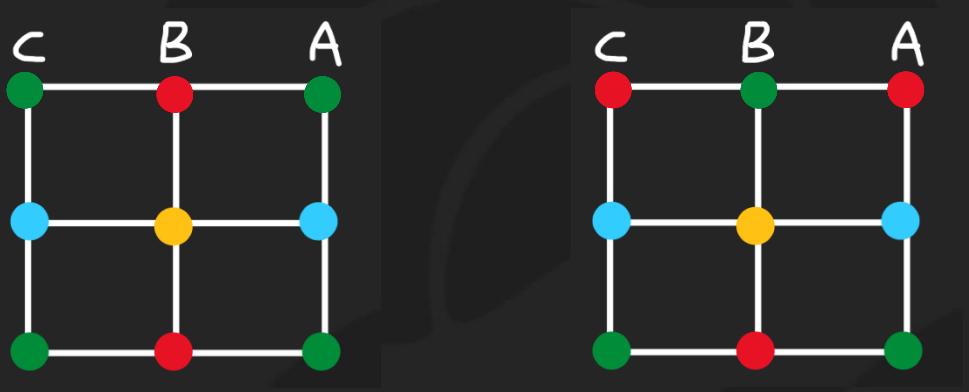
$1 \leq M, N \leq 10^5$

Tinjau contoh pewarnaan pada grid 2×2 yang sudah dibahas diatas

Pola 1



Pola 2



Apapun pola pewarnaanya, setidaknya salah satu dari pernyataan berikut memenuhi

1. Hanya digunakan 2 warna untuk setiap kolomnya

2. Hanya digunakan 2 warna untuk setiap barisnya

Perhatikan jika sebuah baris hanya terdiri dari 2 warna, maka misal urutan warna pada baris ke 1 adalah ... $ababab$ ... dan baris ke 2 adalah ... $cdcdcd$ ..., maka ada $4!$ untuk emmilih a, b, c, d baris ke 3, 5, 7, ... harus diberi warna ... $bababa$ ... atau ... $ababab$ ...

dan baris 4, 6, 8, … harus diberi warna … $cdcdcd$ … atau … $dcdcdc$ …, Maka dalam kasus ini ada $24 \times 2^{jumlah\ baris-1}$ pewarnaan

Dengan cara yang sama diterapkan untuk kolom yang hanya terdiri dari 2 warna, maka ada $24 \times 2^{jumlah\ kolom-1}$ pewarnaan

Namun ada juga kasus dimana baris dan kolom sama sama terdiri dari 2 warna, maka dihitung menggunakan prinsip inkulsi eksklusi

$$24(2^{m-1} \times 2^{n-1} - 1)$$

Kompleksitas : O(N)

Subsoal 4 (12) poin)

$1 \leq M, N \leq 10^{18}$

Untuk subsoal 4 masih menggunakan formula yang sama ya itu

$$24(2^{m-1} \times 2^{n-1} - 1)$$

Namun diperlukan komputasi $x^y\ mod\ p$ dengan kompleksitas O(log N)

Kompleksitas : O(log N)