Schematics 2021

EDITORIAL

SCHEMATICS

NPC

NATIONAL PROGRAMMING CONTEST

SPONSORED BY

Editorial Penyisihan Schematics NPC Senior

25 September 2021

Problem	Expected Difficulty	Author
Anak Kembar	Medium	Aldo Yaputra Hartono
Bagi Himpunan	Easy-Medium	Vania Rizky J W
Covid-21	Easy	Reyner Fernaldy
Dihantam Pandemi	Hard	Zydhan Linnar Putra
Eksplorasi Mineral	Medium-Hard	Daniel Sugianto
Fun Quiz	Easy	Aldo Yaputra Hartono
Gabung Kata	Easy-Medium	Muhamad Affan
Hitung Jalur	Hard	Muhamad Affan & Daniel Sugianto
Impian Pisi	Medium	Muhamad Affan

Tester

- **Andreas Cendranata**
- **Fadhil Musaad**
- **Haniif Ahmad Jauhari**
- **John Stephanus Peter**
- **Nurlita Dhuha Fatmawati**

Anak Kembar

Pembuat Soal : Aldo Yaputra Hartono

Editorialist : Aldo Yaputra Hartono

Tag : Combinatorics, Dynamic Programming

Expected Difficulty : Medium

Observasi 1 : Asumsikan $P(M,N)$ merupakan banyaknya kombinasi peletakan M telur identik pada N piring identik dengan minimal 1 telur pada tiap piring. Mula-mula letakkan 1 telur pada tiap piring sehingga telur yang tersisa sebanyak $M-N$. Karena sudah memenuhi syarat minimal 1 telur tiap piring, maka sisa telur tersebut dapat didistribusikan pada 1 piring, 2 piring, atau bahkan N piring.

$$P(M,N) = \sum_{K=1}^{N} P(M-N,K)$$

Observasi 2 : Asumsikan $Q(M,N)$ merupakan banyaknya kombinasi peletakan M telur identik pada N piring identik dengan minimal 0 telur pada tiap piring. Dikarenakan minimal 0 telur, maka kita bisa meletakkan M telur tersebut pada N piring, $(N-1)$ piring, atau bahkan pada 1 piring saja.

$$Q(M,N) = \sum_{K=1}^{N} P(M,K)$$

Lakukan *precompute* terlebih dahulu lalu keluarkan hasilnya untuk tiap kasus uji setelah dimodulo 10^9+7.

Kompleksitas $O(T+MN)$.

Bagi Himpunan

Pembuat Soal : Vania Rizky J W

Editorialist : Vania Rizky J W

Tag : Math

Expected Difficulty : Easy-Medium

Sebenarnya soal ini sama saja dengan menyusun K bola merah dan $N-K$ bola biru dengan syarat tidak ada 2 bola merah yang bersebelahan.

Susun dahulu $N-K$ bola biru

M bola merah akan ditaruh di celah celah setiap bola biru

$$\_B_1\_B_2\_ \cdots \_B_{N-K-1}\_B_{N-K}\_$$

Ada $N-K+1$ tempat untuk meletakkan K bola merah.

Maka banyaknya cara penempatan adalah

$$\binom{N-K+1}{K}$$

Untuk menghitung nilai kombinasi, kita dapat melakukan prekomputasi untuk nilai bilangan-bilangan faktorial serta nilai *inverse modulo* dari bilangan-bilangan faktorial tersebut. Hal ini dapat dilakukan dalam kompleksitas $O(N)$ atau $O(N \log N)$ tergantung dari implementasi.

Kompleksitas : $O(N)$ atau $O(N \log N)$.

Covid-21

Pembuat Soal : Reyner Fernaldi

Editorialist : Reyner Fernaldi

Tag : Adhoc,String

Expected Difficulty : Easy

Untuk menyelesaikan soal ini, kita dapat menginisasi sebuah array dengan ukuran 10000, $arr[10000]$. Untuk setiap varian virus, dapatkan substring yang terdiri dari 2 sampai 4 karakter. Namun untuk substring yang diawali angka 0 tidak akan di proses. Jadikan substring sebagai indeks pada array $arr[]$. Kemudian, tambahkan *value* array dengan populasi terinfeksi, sesuai indeks (substringnya).

$$\boldsymbol{arr[substring] \mathrel{+}= populasi\ infeksi\ varan\ virus}$$

Sebagai contoh

B1234 50

Substring dari 1234 adalah 12, 23, 34, 123, 234, 1234. Maka,

$arr[12] \mathrel{+}= 50$

$arr[23] \mathrel{+}= 50$

$arr[34] \mathrel{+}= 50$

$arr[123] \mathrel{+}= 50$

$arr[234] \mathrel{+}= 50$

$arr[1234] \mathrel{+}= 50$

Setelah itu kita dapat mencari jawaban dari nilai arr yang telah dihitung

Komplesitas: $O(N)$.

Dihantam Pandemi

Pembuat Soal : Zydhan Linnar Putra

Editorialist : Zydhan Linnar Putra

Tag : Math, Graph, Data Structures

Expected Difficulty : Hard

Observasi : Graf memiliki *weight* yang berbeda sehingga untuk mencari jarak terdekat dapat menggunakan algoritma *dijkstra*.

Pertama, cari banyak *shortest path* dari a dan b ke setiap *node* lainnya yang masing-masing berikutnya akan disebut sebagai variabel $cntShortestPathFromStart$ dan $cntShortestPathFromDst$.

Untuk mendapatkan banyak shortest path dari a ke b yang melewati $node_i$ dapat menggunakan rumus berikut:

$$cntShortestPath_i = cntShortestPathFromStart_i * cntShortestPathFromDst_i$$

Hasil $cntShortestPath_i$ adalah jawaban untuk satu kali perjalanan pulang atau pergi sehingga untuk memenuhi aturan soal hasil tersebut harus dikalikan dua.

Berikutnya, untuk mendapatkan rata-rata kunjungan setiap bulannya hanya perlu membagi hasil yang telah dikali dua tersebut dengan total *shortest path* dari a ke b seperti pada penjelasan contoh masukan dan keluaran soal.

Eksplorasi Mineral

Author : Daniel Sugianto

Editorialist : Muhamad Affan

Tag : Data structures, Dynamic Programming

Expected Difficulty : Medium-Hard

Pertama-tama, mari selesaikan soal ini untuk nilai $D = 1$. Jika D bernilai 1 maka soal ini dapat direduksi menjadi permasalahan untuk mencari luas sub-matriks persegi panjang terbesar yang seluruhnya berisikan nilai 1.

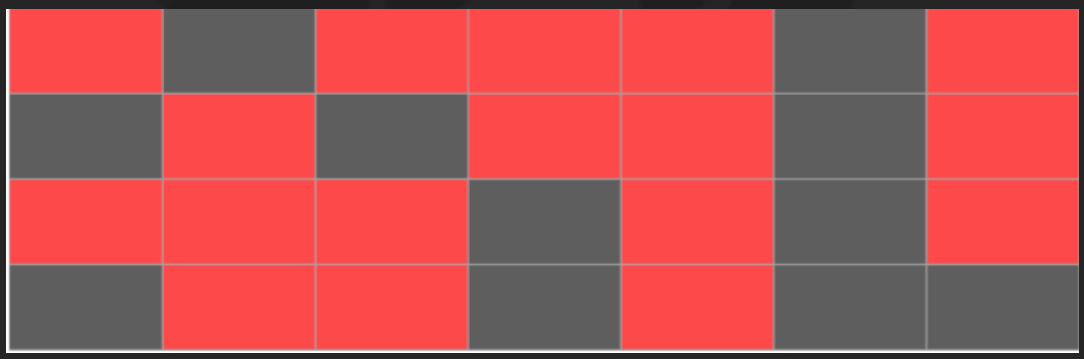
Misalkan matriks awal yang diberikan adalah M.

Kita dapat menggeneralisir solusi ini untuk nilai $D > 1$. Kita dapat membangun matriks $sub_{i,j}[H][L]$ dimana $sub_{i,j}[R][C]$ bernilai 1 jika nilai $M[i][R][C]$ sampai $M[j][R][C]$ semua bernilai 1, dan 0 jika sebaliknya.

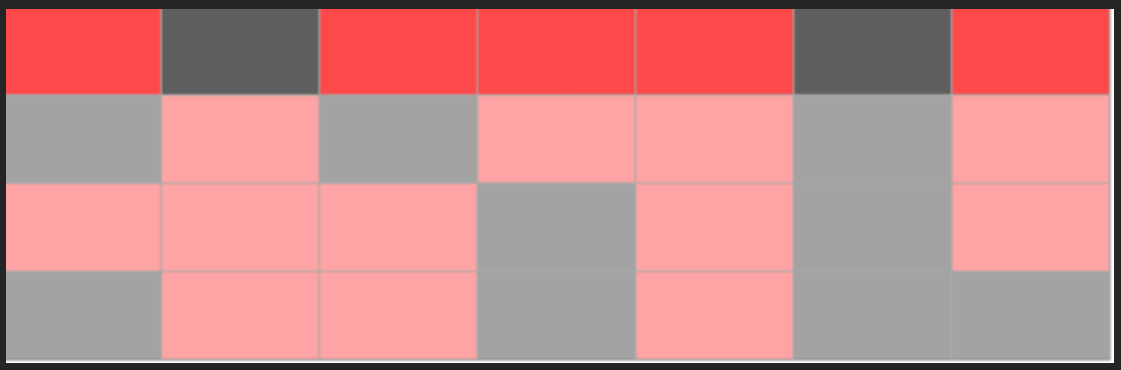
Dengan begitu, kita bisa mendapatkan jawaban dengan mencari luas sub-matriks persegi panjang terbesar pada $sub_{i,j}$ untuk $1 \leq i \leq j \leq D$.

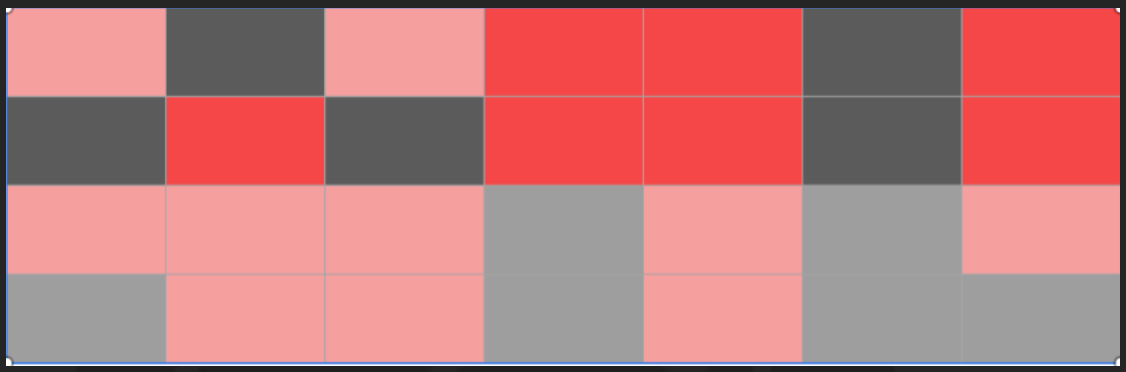
Kita dapat melihat masing-masing baris pada matriks sub sebagai sebuah histogram. Misalkan batang histogram ke-j untuk baris i adalah $Hist[i][j]$. $Hist[i][j]$ bernilai 0 jika $sub[i][j]$ bernilai 0, dan $Hist[i][j]$ bernilai $Hist[i-1][j] + 1$ jika sebaliknya.

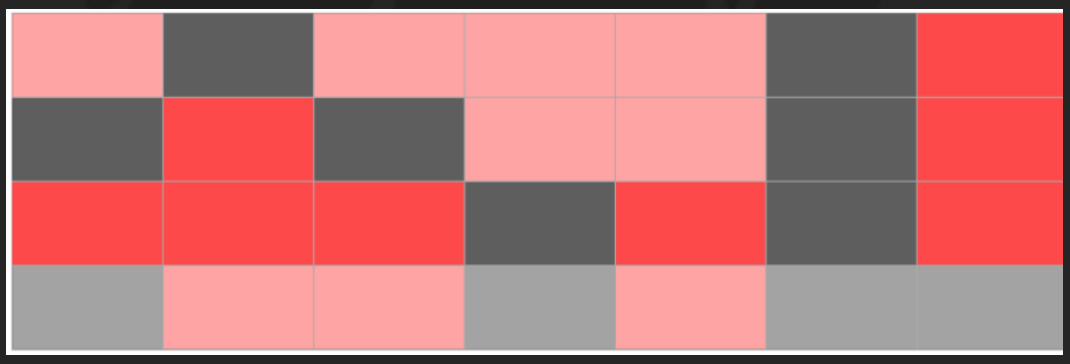
Sebagai contoh, misalkan kotak berwarna merah bernilai 1, serta kotak berwarna hitam bernilai 0. Kita dapat memvisualisasikan matriks seperti ini :

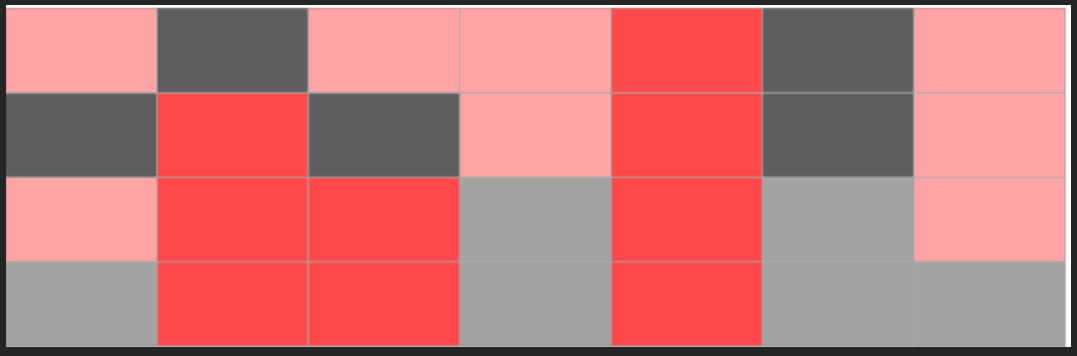


Serta untuk histogram masing-masing baris dapat divisualisasikan seperti ini :









Idenya adalah untuk setiap histogram i, kita akan menghitung histogram terbesar dengan histogram i menjadi histogram terkecil. Untuk itu, kita dapat mencari indeks terdekat yang berada masing-masing di sebelah kiri dan kanan dari i. Misalkan kedua indeks tersebut masing-masing adalah j_L dan j_R, maka histogram terbesar yang dapat dibentuk adalah $(j_R - j_L - 1) * hist[i]$.

Untuk mendapatkan nilai j_L dan j_R untuk setiap i dapat kita lakukan dengan bantuan struktur data *stack* dengan langkah-langkah berikut :

1. Lakukan iterasi untuk histogram i dari $1 - L$.
2. Jika *stack* dalam keadaan kosong, masukan nilai i kedalam *stack*.
3. Jika *stack* tidak dalam keadaan kosong, lakukan operasi *pop* pada *stack* selama nilai histogram pada *top stack* lebih besar dari nilai histogram i. Untuk setiap histogram yang di-*pop*, kita bisa mendapatkan nilai j_L adalah nilai yang berada tepat dibawah nilai yang di-*pop* tadi, serta nilai j_R adalah i.

Jawaban bisa didapatkan dari nilai luas histogram terbesar untuk masing-masing histogram i.

Kompleksitas : $O(D^2LH)$.

Fun Quiz

Pembuat Soal : Aldo Yaputra Hartono

Editorialist : Aldo Yaputra Hartono

Tag : Math

Expected Difficulty : Easy

Observasi 1 : Poin terkecil akan didapat jika semua mata dadu yang muncul bernilai 1. Jika ada N dadu, maka poin terkecilnya adalah $1 \times N = N$.

Observasi 2 : Poin terbesar akan didapat jika semua mata dadu yang muncul bernilai S. Jika ada N dadu, maka poin terkecilnya adalah $S \times N = SN$.

Observasi 3 : Lakukan penjumlahan dari N hingga SN dengan rumus deret aritmatika.

$$\sum_{i=1}^{SN} i - \sum_{j=1}^{N-1} j$$

Jangan lupa keluarkan hasil dalam modulo $10^9 + 7$.

Kompleksitas $O(1)$.

Gabung Kata

Pembuat Soal : Muhamad Affan

Editorialist : Muhamad Affan

Tag : String

Expected Difficulty : Easy-Medium

WLOG, kita asumsikan $|S| < |T|$. Jika S merupakan *substring* dari T, maka jawabannya adalah $|T|$.

Selain itu, kita dapat mencari nilai K terbesar dimana $S_{|S|-K+1}\ S_{|S|-K+2} S_{|S|-K+3} \dots S_{|S|} = T_1 T_2 T_3 \dots T_K$ dan jawabannya adalah $|S| + |T| - K$.

Kedua kasus itu dapat dilakukan dengan menggunakan *Hashing* atau *Prefix Function*.

Kompleksitas : $O(N)$.

Hitung Jalur

Pembuat Soal : Muhamad Affan

Editorialist : Muhamad Affan

Tag : Graph, Data Structure

Expected Difficulty : Hard

Observasi : Dalam sebuah *path* dari verteks U menuju ke verteks V, jika kita melewati sebuah *cycle*, maka jumlah *path* berbeda akan bertambah menjadi 2 kali lipat. Atau dengan kata lain jumlah *path* berbeda dari verteks U ke verteks V adalah 2^x dimana x merupakan jumlah *cycle* yang dilewati dari verteks U menuju verteks V.

Tetapi, mencari jumlah *cycle* secara naif tidak cukup cepat untuk menyelesaikan soal ini. Untuk itu, kita dapat men-dekomposisi graf yang diberikan menjadi sebuah *biconnected component atau block-cut tree*. Dengan begitu, sebuah *cycle* pada graf akan direpresentasikan sebagai sebuah *Block Vertex* pada *block-cut tree* yang telah dibangun.

Sekarang, kita dapat me-*reduce* permasalahan menjadi menghitung ada berapa jumlah *block vertex* dalam *path* dari U ke V. Ini dapat diselesaikan dengan memilih salah satu verteks sebagai *root* dari *tree* lalu melakukan *dfs* untuk menghitung nilai $sum[u]$ untuk semua verteks. Dimana $sum[u]$ adalah jumlah *block vertex* pada *path* dari *root* menuju ke verteks U.

Misalkan $par[u]$ adalah *parent* dari verteks U pada *tree*, serta L adalah *lowest common ancestor* dari U dan V, maka jumlah *block vertex* pada *path* U ke V adalah $sum[u] + sum[v] - sum[L] - sum[par[L]]$. Kita dapat mencari *lowest common ancestor* dari kedua verteks dengan kompleksitas $O(\log N)$ menggunakan teknik *binary lifting* .

Total kompleksitas : $O(Q \log N)$.

Impian Pisi

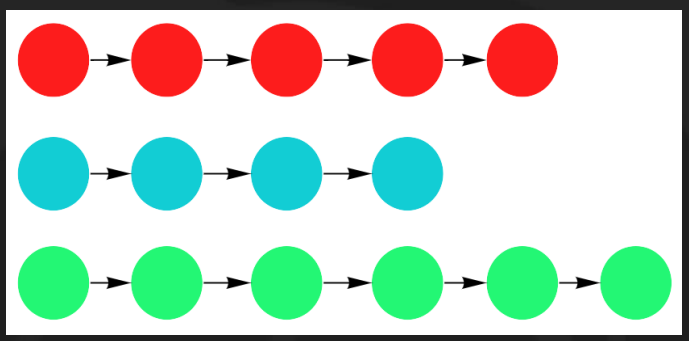
Pembuat Soal : Muhamad Affan & Daniel Sugianto

Editorialist : Muhamad Affan

Tag : Graph, Data Structure

Expected Difficulty : Medium

Observasi 1 : Graf yang diberikan berbentuk beberapa *connected component*, dimana setiap *connected component* berbentuk sebuah *Directed Acyclic Graph* yang tidak memiliki cabang.



Misalkan kita definisikan sebuah *connected component* sebagai sebuah *chain*. Serta posisi verteks *u* pada *chain*-nya adalah pos_u .

Untuk setiap *query* tipe 2, kita dapat membaginya menjadi 2 kasus :

1. *u* dan *v* berada pada *chain* yang sama dan $pos_u \leq pos_v$. Maka jawaban dari *query* adalah 0 karena *v* dapat dicapai dari *u* menggunakan *edge-edge* yang ada.
2. *u* dan *v* berada di *chain* yang berbeda atau *u* dan *v* berada di *chain* yang sama tetapi $pos_u > pos_v$. Misalkan ukuran *chain u* adalah $size_u$, dan $C_x[y]$ adalah nilai C dari verteks di posisi y pada *chain* x.
   Maka jawaban bisa didapatkan dari $\min(C_u[pos_u], C_u[pos_u + 1], C_u[pos_u + 2], \dots, C_u[size_u]) + \min(C_v[1], C_v[2], C_v[3], \dots, C_v[pos_v])$

Untuk dapat men-*support* operasi *query* 1 dan *query* 2 dengan efisien kita dapat membangun *segment tree* untuk setiap *chain* agar proses *update* dan *query* dapat dilakukan dalam kompleksitas $O(\log N)$

Total kompleksitas : $O(N + Q(\log N))$.