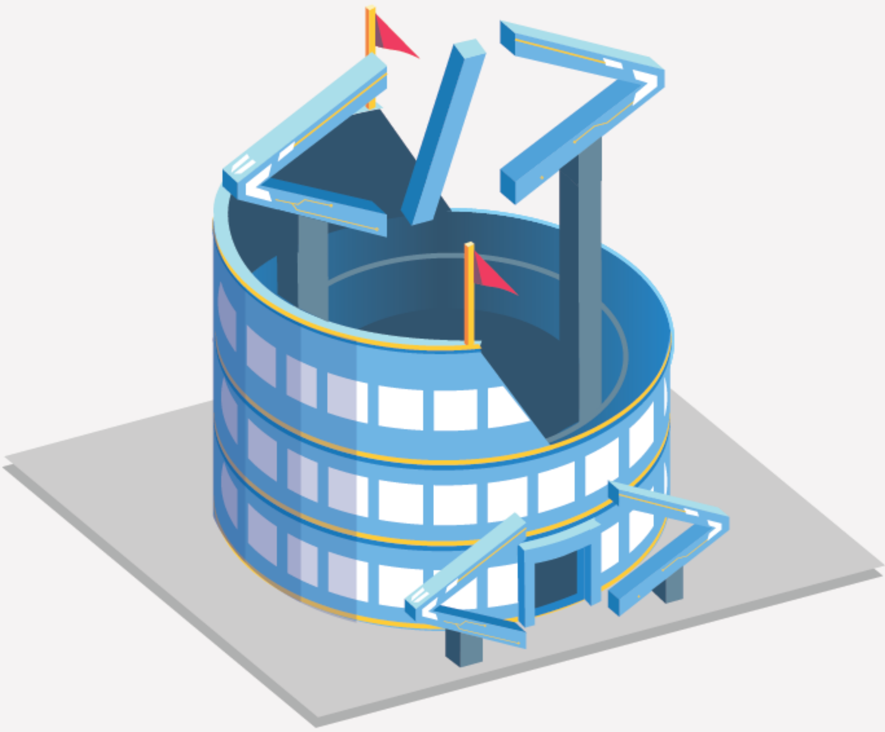
COMPFEST 8

COMPETITION

COMPETITIVE PROGRAMMING CONTEST



Pembahasan Coderclass

SCPC – Minggu 3

Daftar Soal

A. Flow

Tag

complete search

Pembahasan

Soal ini dapat diselesaikan dengan *complete search.* Anggap terdapat C warna yang dinomori $1, 2, \ldots, C$. *Complete search* dilakukan dengan cara mencoba mencari jalur untuk warna 1, lalu mencoba mencari jalur untuk warna 2, dan seterusnya hingga mencari jalur untuk warna C. Jika pernah ditemukan suatu konfigurasi yang valid untuk ke-C warna, maka keluarkan SOLVABLE atau UNSOLVEABLE jika sebaliknya.

Perlu diperhatikan bahwa *complete search* yang dilakukan perlu dilengkapi dengan beberapa *pruning* sederhana, seperti tidak mencoba menempatkan suatu warna di atas petak yang telah berwarna, meletakkan warna namun tidak terhubung dengan sumber warna itu sendiri, dll.

B.Naik Turun

Tag

digit DP

Pembahasan

Sebelum membaca pembahasan ini, ada baiknya membaca sekilas diskusi mengenai digit DP yang terdapat pada tautan ini

Pertama, kita definisikan $f(x)$ sebagai banyaknya bilangan galau maksimal pada rentang $[0, x]$. Menggunakan definisi tersebut, tentunya kita bisa mendapatkan jawaban pertanyaan di soal dengan menghitung $f(B) - f(A-1)$. Persoalannya, apakah kita bisa membuat fungsi $f(x)$ tersebut? Ternyata bisa!

Kita akan memanfaatkan teknik *dynamic programming* yang bekerja di digit, biasanya disebut digit DP. Pertama, kita observasi $f(x)$. Misal, x terdiri dari N digit, dan ditulis $x_{N-1} \ldots x_1 x_0$ atau dengan kata lain, $x = \sum_{i=0}^{N-1} x_i \cdot 10^i$. Maka, kita bisa menghitung $f(x)$ menggunakan bantuan DP dengan *state* indeks, lebih/kurang dari digit terakhir, dan digit terakhir, ditambah iterasi. Pada setiap iterasi, kita akan meng-*fix*-kan nilai dari indeks terbesar i, sehingga nilai digit pada indeks tersebut kurang dari x_i, dan untuk tiap $j > i$, nilai digitnya pada indeks ke-j adalah x_j. Kurang lebih, rekurens DP untuk membantu penghitungan adalah sebagai berikut:

$$g(index, stat, last\_digit) = \begin{cases} 1 & , index = -1 \\ \sum_{i=last\_digit+1}^{9} g(index-1, false, i) & , index \geq 0 \land stat = true \\ \sum_{i=0}^{last\_digit-1} g(index-1, true, i) & , index \geq 0 \land stat = false \end{cases}$$

Selanjutnya, kita bisa mencari $f(x)$. Maka, kita perlu melakukan penghitungan 2 kali, yaitu untuk mencari $f(B)$ dan $f(A-1)$. Perhatikan bahwa dalam mencari $A-1$, dibutuhkan sedikit operasi *BigInteger*. Akhirnya, kompleksitas solusi ini adalah $O(logB + logA)$, dengan konstanta yang agak besar.

C. XOR A B

Tag

ad hoc

Pembahasan

Pertama, kita buat dulu $f(x) = 0 \oplus 1 \oplus \ldots \oplus x$. Ini akan mempermudah pencarian solusi nantinya. Namun, bagaimana cara menghitung $f(x)$ dengan cepat?

Untuk mempermudah pekerjaan, kita dapat melakukan perhitungan untuk masing-masing bit. Apabila diamati, nilai *xor* dari bit ke-i, pasti bernilai 0 pada kelipatan 2^{i+1} kecuali pada bit ke-0 (menjadi kasus khusus). Karena itu kita dapat melihat hasil dari bit ke-i dengan menghitung hasil dari $x\ (mod\ 2^{i+1})$, sebut angka tersebut y. apabila $y < 2^i$, maka hasil *xor*-nya pasti 0. Apabila $y - 2^i$ adalah genap, maka nilai *xor* untuk bit tersebut adalah 1, dan 0 apabila ganjil.

Nah, bagaimana cara memanfaatkan $f(x)$ untuk menghitung pada suatu rentang $[A, B]$? Kita bisa memanfaatkan salah satu properti operasi *xor*, yaitu $x \oplus x = 0$. Untuk pertanyaan dengan rentang $[A, B]$, dapat dihitung sebagai $f(B) \oplus f(A-1)$. Ini karena:

$$f(A-1) \oplus f(B)$$
$$= f(A-1) \oplus f(A-1) \oplus A \oplus (A+1) \ldots \oplus (B)$$
$$= 0 \oplus A \oplus (A+1) \ldots \oplus (B)$$
$$= A \oplus (A+1) \ldots \oplus (B)$$

Maka, kita bisa menjawab tiap pertanyaan dengan cepat, mengingat kompleksitas solusi di atas adalah $O(logA + logB)$. Sisanya, yang perlu diperhatikan adalah mengonversinya ke biner.

Dalam pengerjaannya, hati-hati pada bit ke-63, perhatikan setiap operasi agar tidak *overflow*. Selain itu, hati-hati pada kasus dimana $A = 0$.

Selain solusi di atas, ada solusi lain untuk mencari $f(x)$. Hint: perhatikan nilainya untuk setiap modulo 4.

D. Truel

Tag

graf

Pembahasan

Perhatikan bahwa relasi mendominasi-didominasi dalam soal ini dapat dipetakan ke dalam sebuah *tournament graph*. Sehingga, permasalahan dalam soal ini dapat direduksi menjadi mencari tahu apakah terdapat setidaknya sebuah *cycle* dengan panjang 3 (untuk selanjutnya akan disebut sebagai *triangle*) pada graf tersebut.

Pencarian sebuah *triangle* pada graf paling cepat berjalan dalam kompleksitas $\frac{N^3}{64}$ dan itu pun masih mendapat putusan TLE jika digunakan untuk menjawab permasalahan ini.

Lemma 1

Jika terdapat sebuah *cycle* dengan panjang lebih dari tiga pada suatu *tournament graph*, maka pada graf tersebut dapat ditemukan pula *cycle* dengan panjang tepat tiga.

Bukti Lemma 1

Anggap dalam graf tersebut terdapat *cycle* terpendek C (*cycle* terpendek berarti *cycle* dengan banyak $vertex$ tersedikit). Apabila C memiliki panjang tiga, maka telah ditemukan *cycle* dengan panjang tiga. Namun, apabila $|C| > 3$, anggap terdapat tiga buah $vertex$ x, y, z pada *cycle* tersebut sehingga $x \rightarrow y \rightarrow z$ merupakan salah satu *path* dari *cycle* tersebut.

Karena graf merupakan *tournament graph*, antara x dan z pasti terdapat suatu edge/sisi. Apabila sisi tersebut berbentuk $z \rightarrow x$, maka C bukan merupakan *cycle* terpendek (karena terdapat *cycle* $x \rightarrow y \rightarrow z \rightarrow x$). Apabila sisi tersebut berbentuk $x \rightarrow z$, maka C juga bukan merupakan cycle terpendek karena sisi $x \rightarrow z$ dan sisi-sisi lainnya dalam C selain yang melewati y membentuk *cycle* lain yang panjangnya tepat lebih pendek satu *path* dari C sendiri.

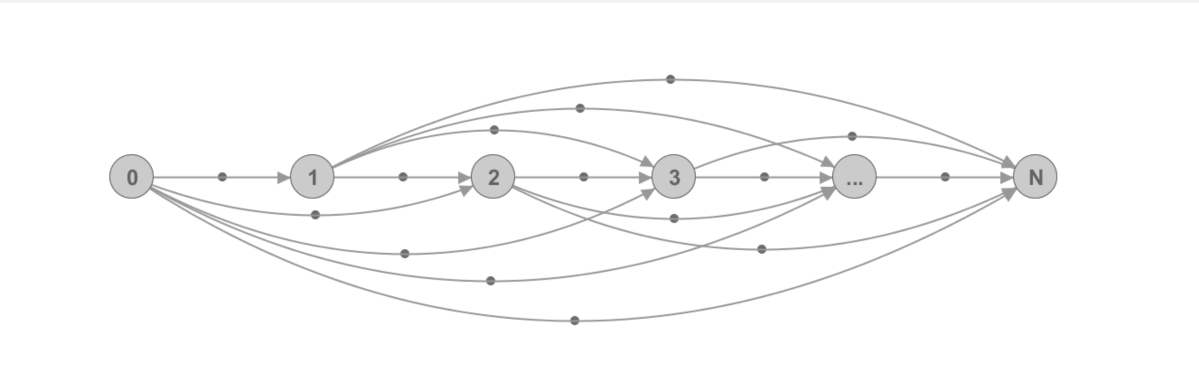
Oleh karena itu, dapat dideduksi bahwa tidak mungkin terdapat suatu *cycle* terpendek C dalam suatu *tournament graph* sehingga $|C| > 3$. Perlu diperhatikan juga bahwa $|C|$ tidak mungkin satu atau dua karena *tournament graph* tidak menginginkan adanya *self-loop* atau *cycle* dalam bentuk $x \rightarrow y \rightarrow x$.

Dari **Lemma 1**, dapat ditarik kesimpulan lagi bahwa permasalahan soal ini dapat direduksi kembali menjadi mencari setidaknya satu *cycle* (panjang berapapun) dari graf yang diberikan.

Lalu, bagaiman mencari *cycle*-panjang-berapapun dari graf yang diberikan?

Permasalahan ini jawabannya dapat dicari dengan mencari kebenaran dari komplemennya: Apakah tidak terdapat *cycle* pada graf yang diberikan?

Perhatikan bahwa *tournament graph* tidak mengandung *cycle* apabila himpunan nilai-nilai *in-degree* dari *vertex-vertex*-nya ekivalen dengan $\{0, 1, 2, \dots, |V| - 1\}$ dengan V adalah himpunan *vertex* dan $|V|$ menyatakan banyaknya *vertex*, atau dengan kata lain, *tournament graph* tersebut juga merupakan *Directed Acyclic Graph (DAG)*. Hal ini benar karena untuk suatu *tournament graph*, hanya ada satu bentuk yang merupakan DAG, yaitu seperti berikut:



Atau dengan kata lain, untuk suatu *tournament graph* yang asiklis dengan banyaknya *vertex* $|V|$:

- Terdapat tepat satu *vertex* v_0 yang memiliki *in-degree* 0: untuk semua vertex $v_i \neq v_0$, terdapat sisi $v_i \to v_0$
- Terdapat tepat satu *vertex* v_1 yang memiliki *in-degree* 1: untuk semua vertex $v_i \neq v_1$, $v_i \neq v_0$, terdapat sisi $v_i \to v_1$
- Dan seterusnya, hingga *vertex* v_n yang memiliki *in-degree* sebesar $|V| - 1$.

Pengecekan nilai *in-degree* dari setiap *vertex* ini dapat dilakukan dalam waktu $O(E)$ atau dengan kata lain $O(V^2)$.

E. Next Balanced Parentheses

Tag

greedy

Pembahasan

Pada pembahasan ini, akan digunakan *1-based indexing*. Misal buka-tutup kurung yang diberikan di soal adalah X, panjang X adalah N, dan buka-kurung tutup yang leksikografis terkecil setelah X adalah Y. Maka, observasinya adalah terdapat sebuah indeks i, sehingga i sebesar mungkin serta:

- Untuk setiap $1 \leq j < i, X_j = Y_j$
- $X_i =$ '(', dan $Y_i =$ ')'

Maka, sebenarnya persoalan ini berubah menjadi: Carilah indeks i, sehingga $X_i =$ '(', dan terdapat suatu konfigurasi buka kurung dan tutup kurung untuk setiap $j > i$, sehingga jika $X'_i =$ ')' , ia menjadi konfigurasi buka-tutup kurung yang valid. Bagaimana cara memeriksanya?

Pertama, kita buat karakter '(' bernilai +1, dan ')' bernilai -1. Misal, didefinisikan P_i adalah *prefix sum* ke-i dari pemberian nilai tersebut. Maka, pada konfigurasi buka-tutup kurung yang valid, nilai $P_i \geq 0$ untuk $1 \leq i \leq N$, serta $P_N = 0$. Perhatikan apa yang terjadi apabila kita mengubah karakter X_i dari '(' menjadi ')'. Tentunya, P_i' bernilai $P_i - 2$. Apabila $P_i' \geq 0$, ternyata pasti terdapat suatu cara menyusun $N - i$ karakter sisanya, sehingga membentuk suatu buka-tutup kurung Y, yang leksikografis terkecil lebih besar dari X. Nah, bagaimana menyusun $N - i$ karakter sisanya?

Pertama, kita catat jumlah dari nilai karakter, simpan dalam variabel yang sebut saja sum. Untuk $i - 1$ karakter pertama, tidak ada pengubahan karakter. Untuk karakter ke-i, karakter berubah dari '(' menjadi ')'. Lalu, untuk menyusun karakter ke-j, $i < j \leq N$, kita lakukan berturut-turut:

- Seandainya $sum + 1 \leq N - j$, maka karakter ke-j adalah '('. Ini karena kita masih bisa "menutup" belakangan.
- Jika tidak, maka karakter ke-j adalah ')'.

Tentunya, sambil memasang karakter, kita juga meng-*update* sum sesuai penambahan karakter yang terjadi.

Perhatikan bahwa ada kasus dimana X adalah buka-tutup kurang yang leksikografis terbesar. Pada kasus tersebut, tentunya tidak terdapat indeks i, $1 \leq i \leq N$, sehingga $P_i \geq 2$ dan $X_i =$ '('. Jika terjadi, maka keluarkan "TIDAK ADA".

F. Variabel Sangat Bebas

Tag

graf, single-source shortest path, constraint graph

Pembahasan

Soal ini dapat diselesaikan dengan cara memodelkan masalah ini menjadi graf. Verteks-verteks dari graf adalah setiap variabel yang ada pada sebuah kasus uji (sebut saja variabel x_1 hingga x_N). Sisi-sisi dari graf adalah sisi-sisi berarah, yang menghubungkan verteks-verteks dari variabel x_{b_i} menuju variabel x_{a_i} dengan bobot r_i. Selain itu, ditambahkan pula sebuah verteks x_0 dengan sisi-sisi keluar menuju semua verteks lainnya.

Sehingga, graf yang terbentuk adalah graf $G = < V, E >$ dengan $V = \{x_0, x_1, \dots, x_N\}$ dan $E = \{(x_{b_i}, x_{a_i}) | terdapat\ batasan\ x_{a_i} - x_{b_i} \leq r_i\} \cup \{(x_0, x_1), (x_0, x_2), \dots, (x_0, x_N)\}$ dengan setiap sisi-nya memiliki bobot sesuai r_i atau 0 bila sisi tersebut keluar dari x_0.

Sebut $d(x_0, x_k)$ sebagai jarak terdekat (*shortest-path*) dari verteks x_0 menuju verteks x_k. Maka, salah satu solusi yang valid untuk permasalahan pada soal ini adalah $x_k = d(x_0, x_k)$ untuk setiap $k \in [1, N]$.

Mengapa hal ini bisa benar?

Perhatikan bahwa *triangle inequality* berlaku pada *shortest path*, yaitu untuk tiga verteks berbeda a, b, dan c dan terdapat sisi dari b ke c dengan bobot $w(b, c)$ dan *shortest path* dari a ke b adalah $d(a, b)$ dan dari a ke c adalah $d(a, c)$, berlaku pertidaksamaan berikut:

$$d(a, c) \leq d(a, b) + w(b, c)$$

Maka, jika terdapat batasan-batasan $x_{a_i} - x_{b_i} \leq r_i \equiv x_{a_i} \leq x_{b_i} + r_i$ dan hal tersebut direpresentasikan dalam graf seperti pada penjelasan sebelumnya serta nilai dari $x_{a_i} = d(x_0, x_{a_i})$ dan $x_{b_i} = d(x_0, x_{b_i})$, *triangle inequality* terpenuhi:

$$x_{a_i} \leq x_{b_i} + r_i$$

$$d(x_0, x_{a_i}) \leq d(x_0, x_{b_i}) + w(x_{b_i}, x_{a_i})$$

Namun, solusi di atas tidak dapat diselesaikan menggunakan algoritme *shortest-path* Dijkstra dikarenakan keterbatasannya mendeteksi *negative cycle*. Perhatikan kasus apabila terdapat batasan-batasan berikut:

$$x_1 - x_2 \leq -1$$

$$x_2 - x_3 \leq -1$$

$$x_3 - x_1 \leq -1$$

Constraint graph untuk ketiga pertidaksamaan berikut membentuk *cycle* yang memiliki total bobot negatif, sehingga jika algoritme Dijkstra dijalankan pada graf tersebut, dapat terjadi *infinite loop*.

Oleh karena itu, solusi yang diharapkan untuk soal ini adalah menggunakan algoritma Bellman-Ford, yang dapat mendeteksi adanya *negative cycle*.

G. Memotong Kue

Tag

geometri, matematika

Pembahasan

Pandang setiap titik potong dan titik sudut segitiga dan segmen-segmen garis yang menghubungkannya sebagai graf planar. Perhatikan bahwa ada 4 jenis verteks pada graf ini berdasarkan derajatnya:

- Verteks berderajat $N+2$, sebanyak 3 verteks, yaitu ketiga titik sudut segitiga,
- Verteks berderajat 3, sebanyak $3N$ verteks pada sisi-sisi segitiga,
- Verteks berderajat 4, misalkan sebanyak α, terletak di dalam segitiga yang dilalui oleh tepat 2 garis potongan kue, dan
- Verteks berderajat 6, misalkan sebanyak β, terletak di dalam segitiga yang dilalui oleh tepat 3 garis potongan kue.

Dari klasifikasi ini, kita peroleh $|V| = 3 + 3N + \alpha + \beta$ dan berdasarkan Handshake lemma, juga berlaku $2|E| = 3(N+2) + 3(3N) + 4\alpha + 6\beta$ sehingga $|E| = 6N + 3 + 2\alpha + 3\beta$.

Berdasarkan rumus Euler untuk graf planar, berlaku

$$|F| = |E| - |V| + 2$$

Sehingga diperoleh $|F| = 3N + 2 + \alpha + 2\beta$. Jawaban yang diinginkan adalah $|F| - 1$, yaitu $3N + 1 + \alpha + 2\beta$.

Jika ditarik $3N$ garis potong tanpa ada tiga garis potong yang bertemu di satu titik, maka akan terdapat sebanyak $3N^2$ titik. Namun, untuk setiap tiga garis potong yang berpotongan di satu titik, maka sebanyak dua titik berkurang, sehingga akan tersisa graf planar akhir, yaitu sebanyak $\alpha + \beta$ verteks. Dengan demikian, diperoleh hubungan $3N^2 - 2\beta = \alpha + \beta$ sehingga $\alpha = 3N^2 - 3\beta$.

Dengan demikian, jawaban yang diharapkan adalah $1 + 3N + 3N^2 - \beta$. Untuk mencari β, yaitu banyaknya titik konkurensi tiga buah garis potong, kita memerlukan geometri dan iterasi.

Misalkan ABC adalah segitiga sama sisi untuk kue Pak Chanek dan misalkan A_i, B_i, C_i $(1 \leq i \leq N)$ adalah titik-titik berjarak sama pada sisi BC, CA, AB berturut-turut. Tanpa mengurangi keumuman, panjang sisi kue Pak Chanek adalah $N + 1$ sehingga jarak antar titik penanda adalah 1 satuan.

Terdapat kemungkinan pemotongan AA_i, BB_j, CC_k konkuren pada satu titik. Kita ingin menghitung terlebih dahulu banyaknya konkurensi. Kita memerlukan observasi geometris berikut.

Tiga potongan AA_i, BB_j, CC_k akan konkuren jika dan hanya jika

$$\frac{BA_i}{A_iC} \cdot \frac{CB_j}{B_jA} \cdot \frac{AC_k}{C_kB} = 1$$

Sifat ini dapat diperoleh dengan memanfaatkan perbandingan luas segitiga dengan tinggi yang sama dengan perbandingan alasnya (lihat Ceva's theorem).

Dengan demikian, kita cukup mencari banyaknya tripel (i, j, k) dengan $1 \leq i, j, k \leq N$ sehingga

$$ijk = (N + 1 - i)(N + 1 - j)(N + 1 - k)$$

Untuk memangkas kompleksitas, persamaan di atas ekivalen dengan

$$k = \frac{(N + 1)(N + 1 - i)(N + 1 - j)}{(N + 1 - i)(N + 1 - j) + ij}$$

dimana ekspresi di kanan pasti tidak lebih dari $N + 1$, sehingga jika ekspresi kanan bilangan asli pasti bernilai 1 hingga N.

Jadi, cukup kita cari banyak pasangan (i, j) sehingga penyebut membagi pembilang pada sisi kanan persamaan terakhir.

Dengan demikian, β sama dengan banyaknya pasangan (i, j) dengan $1 \leq i, j \leq N$ sehingga $(N + 1 - i)(N + 1 - j) + ij$ habis membagi $(N + 1)(N + 1 - i)(N + 1 - j)$. Banyaknya pasangan tersebut dapat dicari dengan kompleksitas waktu $O(N^2)$.

H. Jus Tropis

Tag

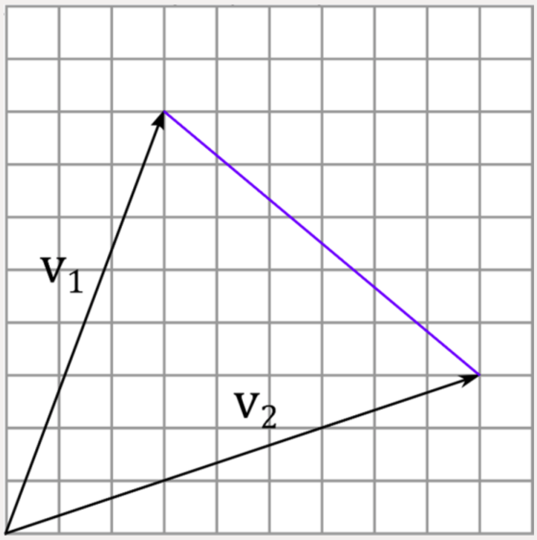
geometri (convex hull)

Pembahasan

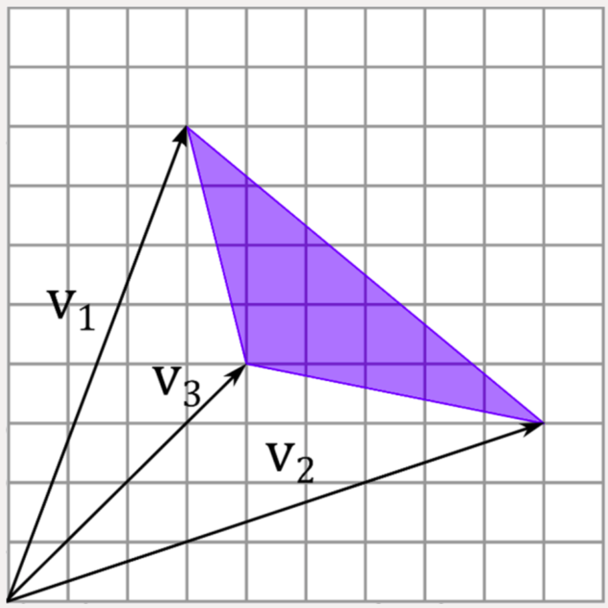
Anggap seluruh jus dengan komposisi P pisang dan Q mangga adalah vektor-vektor dalam ruang vektor 2D: Jus ke-i adalah vektor $v_i = \langle P_i, Q_i \rangle$. Maka, solusi untuk suatu pertanyaan: Apakah dapat dibentuk jus dengan komposisi R pisang dan S mangga, adalah dengan memeriksa apakah titik ujung vektor $w = \langle R, S \rangle$ terdapat dalam *convex hull* yang dibentuk oleh titik-titik ujung vektor-vektor v_i.

Mengapa hal tersebut bisa benar?

Anggap hanya terdapat dua jus pada awalnya, yaitu $v_1 = \langle P_1, Q_1 \rangle$ dan $v_2 = \langle P_2, Q_2 \rangle$. Dari kedua vektor tersebut, vektor-vektor lainnya yang dapat dibentuk oleh kombinasi kedua vektor tersebut pasti memiliki bentuk $x\langle P_1, Q_1 \rangle + (1-x)\langle P_2, Q_2 \rangle$. Secara geometris, penggambarannya adalah sebagai berikut. Garis biru menyatakan daerah vektor-vektor $w = x\langle P_1, Q_1 \rangle + (1-x)\langle P_2, Q_2 \rangle$.



Jika terdapat tiga vektor, berikut penggambarannya secara geometris. Daerah yang diarsir biru merupakan daerah vektor yang dapat dinyatakan dalam kombinasi linear v_1, v_2, v_3.



Hal yang sama berlaku hingga tak banyak titik.

Sehingga, benar bahwa solusinya adalah dengan membuat *convex hull* dari ke-N titik-titik masukan, kemudian untuk setiap ke-M pertanyaan, cek apakah titik pertanyaan itu berada di dalam *convex hull* yang telah dibuat tersebut. Pengecekan dapat dilakukan dalam $O(N)$.

Kompleksitas waktu yang diharapkan dari solusi adalah $O(N \log N + NM)$.