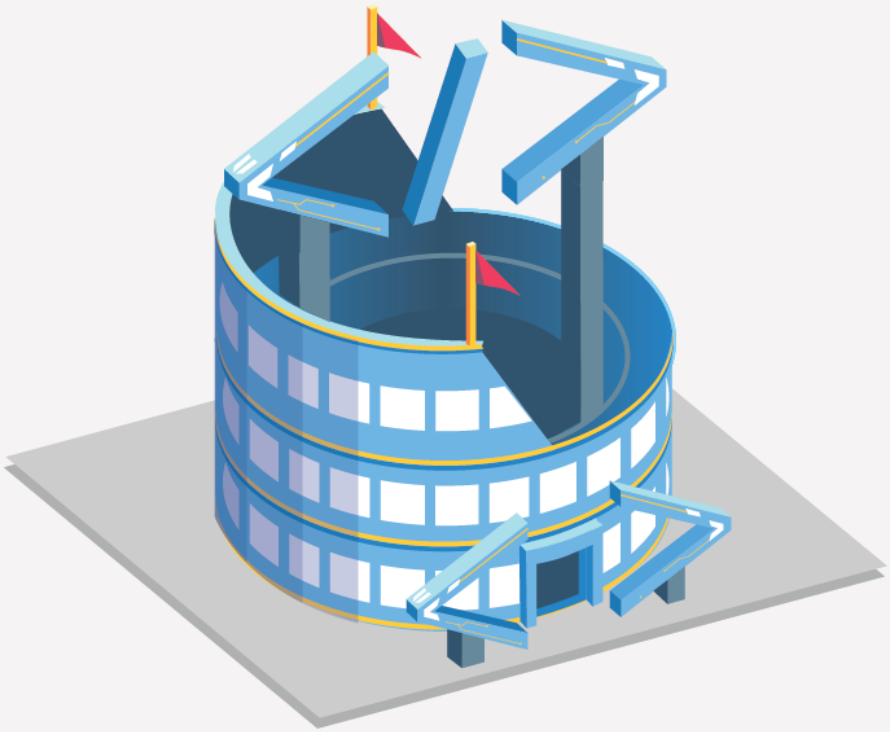
COMPFEST 8

COMPETITION

COMPETITIVE PROGRAMMING CONTEST



COMPFEST 8

Pembahasan Coderclass

SCPC – Minggu 4

Daftar Soal

A. Bukan Convex Hull
B. Channek Pen sCanner
C. Count ST
D. ICP Company
E. Power Rancher
F. Pusing Pusing Poligon
G. Segmen Sirkular
H. Tumbuh-Tumbuhan

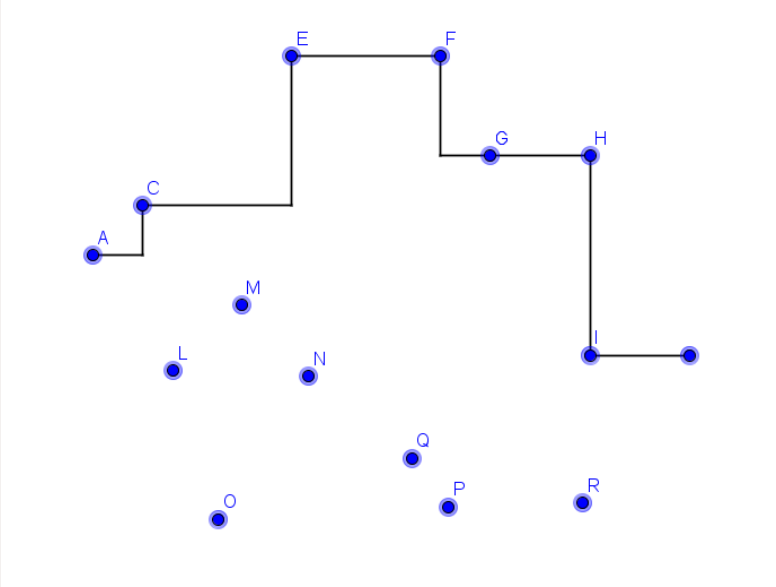
A. Bukan Convex Hull

Tag

line sweep, stack

Pembahasan

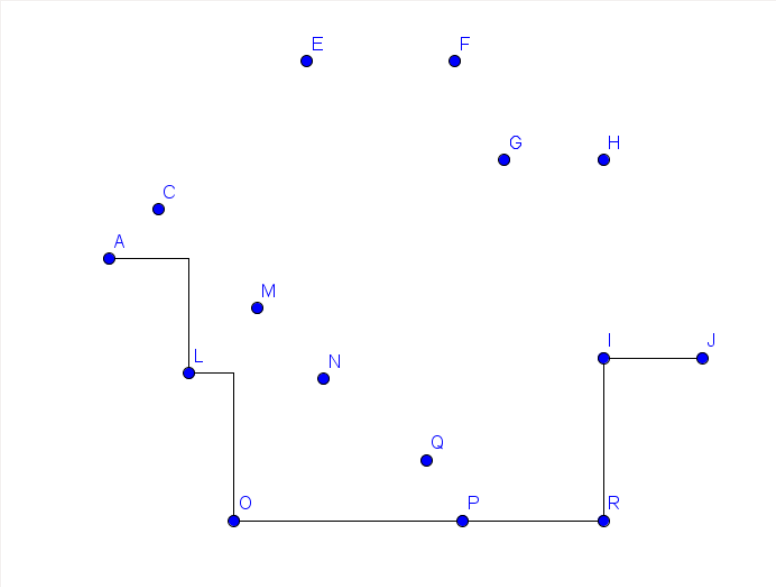
Pertama, kita memulai dengan membuat *hull* bagian atas. *Hull* ini dimaksudkan untuk membentuk batas atas dari poligon yang akan kita buat. Perhatikan bahwa *hull* bagian atas yang kita buat akan menyerupai gunung (menaik, kemudian turun). *Hull* ini dapat dibuat dengan melakukan *line sweep*. Cara yang digunakan mirip dengan algoritma *Graham's scan* untuk mencari *convex hull*. Mula-mula kita urutkan semua titik berdasarkan nilai x, seandainya seri, berdasarkan y yang lebih kecil. *Hull* disini direpresentasikan sebagai sebuah *stack*. Setiap kali kita mencoba memasukan sebuah titik kedalam *hull* kita, mula-mula kita periksa apakah bila titik ini kita masukan, *hull* kita tetap valid (berbentuk gunung, yaitu naik, kemudian turun). Apabila *hull* kita tetap valid, maka masukkan titik tersebut. Apabila tidak (misalnya mengakibatkan naik, turun, lalu naik), maka kita akan melakukan *pop* pada *hull*, dan lakukan hingga *hull* yang kita miliki valid dan kita dapat memasukan titik tersebut.



contoh hull bagian atas

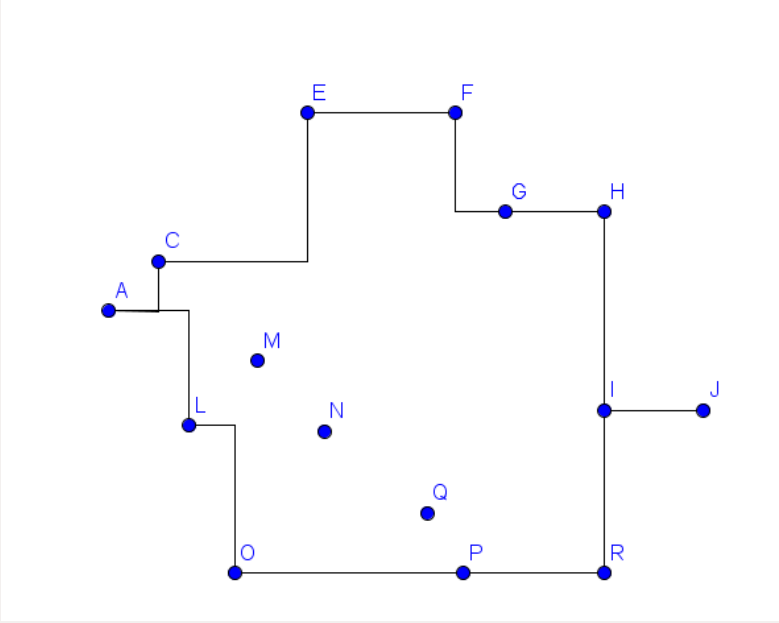
Setelah itu, kita akan membuat *hull* bagian bawah. *Hull* ini dimaksudkan untuk membentuk batas bawah dari poligon yang kita buat. Perhatikan bahwa hull bagian bawah yang

kita buat akan menyerupai lembah (menurun, kemudian naik). *Hull* ini dapat dibuat dengan cara yang sama dengan *hull* bagian atas, hanya saja dibalik.



contoh hull bagian bawah

Setelah kita mendapatkan *hull* bawah dan *hull* atasnya, kita dapat menggabungkan kedua *hull* tersebut dan membentuk sebuah poligon yang memenuhi syarat dan dengan luas minimal.



Contoh penggabungan hull bagian atas dan bawah

Perhatikan bahwa saat kita membuat hull bagian bawah, kita juga harus memperhatikan garis yang dibuat saat membuat hull bagian atas agar tidak terjadi garis yang saling berpotongan.

Total kompleksitas untuk solusi ini adalah $O(N \log N)$.

B. Channek Pen sCanner

Tag

ad hoc

Pembahasan

Kita cukup memeriksa sesuai deskripsi soal. Pada saat memeriksa, kita bisa memeriksa beberapa karakter saja yang bisa membedakan ‘A’, ‘B’, ‘C’, dan spasi untuk mempercepat pemeriksaan.

Bagian yang cukup *tricky* pada soal ini adalah batasan 150.000 huruf. Ingat, 150.000 huruf berarti membutuhkan panjang 600.000 karakter. Hal ini bisa menjebak peserta yang menggunakan *array of char* untuk menyimpan string masukan.

C. Count ST

Tag

cactus graph, DFS spanning tree

Pembahasan

Apabila diamati, dalam pembuatan *spanning tree*, untuk setiap *cycle*, kita harus membuang salah satu edge pada *cycle* tersebut. Tentunya, untuk suatu *cycle* berukuran C, maka ada C cara memilih edge yang bisa dibuang. Mengingat pembuangan edge antar *cycle* bersifat independen, banyak cara adalah dengan mengalikan cara antar *cycle*. Misal, *cycle-cycle* yang terbentuk adalah $C_1, C_2, \ldots, C_k$. Maka, banyak caranya adalah:

$$\prod_{i=1}^{k} |C_i|$$

Sekarang, yang menjadi pertanyaan adalah, bagaimana cara mencari ukuran tiap *cycle* dengan efisien?

Kita bisa membuat sebuah *rooted tree* dengan menggunakan *depth-first search*, yang biasa disebut DFS *spanning tree*. Bagaimana cara memanfaatkan *tree* ini? Awalnya, kita ambil vertex sembarang sebagai *root*. Lalu, jalankan DFS, dan catat kedalaman tiap vertex (anggap *root* memiliki kedalaman 0). Observasinya, setiap edge yang terdapat dalam graf awal, namun tidak terdapat dalam *tree*, pasti merupakan *back edge*, yang termasuk dalam tepat satu *cycle*. Observasi selanjutnya, panjang dari *cycle* tersebut pasti merupakan selisih antara kedalaman 2 vertex yang edge tersebut hubungkan ditambah 1. Maka, kita bisa mendapatkan seluruh ukuran *cycle* dengan cepat.

Kompleksitas akhir untuk solusi ini adalah $O(N + M)$.

D. ICP Company

Tag

möbius inversion formula, line sweep

Pembahasan

Untuk mengerjakan soal ini, dibutuhkan pemahaman mengenai *möbius function* dan *möbius inversion formula.* Kedua topik tersebut dapat dibaca di tautan-tautan berikut:

- https://en.wikipedia.org/wiki/M%C3%B6bius_function
- https://en.wikipedia.org/wiki/M%C3%B6bius_inversion_formula
- https://discuss.codechef.com/questions/72953/a-dance-with-mobius-function

Möbius inversion formula menyatakan bahwa jika g dan f adalah fungsi aritmatika, dimana

$$g(n) = \sum_{d|n} f(d) \ \ for\ every\ integer\ n \geq 1$$

maka

$$f(n) = \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right) \ for\ every\ integer\ n \geq 1$$

Kita akan menggunakan properti tersebut untuk menyelesaikan soal ini. Kita akan membongkar rumus yang di soal seperti berikut:

$$\sum_{a=1}^{A} \sum_{b=a+1}^{B} \frac{b-a}{\gcd(a,b)}$$

$$= \sum_{a=1}^{A} \sum_{b=a+1}^{B} (b-a) f(\gcd(a,b))$$

$$= \sum_{a=1}^{A} \sum_{b=a+1}^{B} (b-a) \sum_{d|\gcd(a,b)} g(d)$$

$$= \sum_{d=1}^{A} \sum_{i=1}^{\lfloor \frac{A}{d} \rfloor} \sum_{j=i+1}^{\lfloor \frac{B}{d} \rfloor} (di - dj) g(d)$$

$$= \sum_{d=1}^{A} d \cdot g(d) \sum_{i=1}^{\lfloor \frac{A}{d} \rfloor} \sum_{j=i+1}^{\lfloor \frac{B}{d} \rfloor} (i-j)$$

$$= \sum_{d=1}^{A} \sum_{x|d} x \cdot \mu(x) \sum_{i=1}^{\lfloor \frac{A}{d} \rfloor} \sum_{j=i+1}^{\lfloor \frac{B}{d} \rfloor} (i-j)$$

Perhatikan bahwa nilai dari $\sum_{x|d} x \cdot \mu(x)$ sebenarnya bisa di prekomputasi menggunakan teknik sejenis *sieve*, yang berjalan dalam $O(A \log A)$ karena *harmonic series*. Sisanya, penghitungan $\sum_{d=1}^{A} \sum_{x|d} x \cdot \mu(x) \sum_{i=1}^{\lfloor \frac{A}{d} \rfloor} \sum_{j=i+1}^{\lfloor \frac{B}{d} \rfloor} (i-j)$ dapat dilakukan dengan menggunakan *line sweep*, yang dibentuk dengan $O(\sqrt{A})$ *event*, dimana mengurutkannya butuh $O(\sqrt{A} \log A)$. Maka, kompleksitas akhir untuk menyelesaikannya adalah $O(A \log A)$ prekomputasi, dan $O(\sqrt{A} \log A)$ per kasus untuk *line sweep*.

E. Power Rancher

Tag

meet in the middle, struktur data

Pembahasan

Untuk mempermudah pengerjaan soal ini, kita bisa melakukan dilatasi, yaitu mengalikan tiap koordinat dengan 4. Hal ini agar kita bisa melakukan semua pekerjaan dalam bilangan bulat, karena kita seperti menghilangkan pembagian dengan 4. Perhatikan, pada soal maupun pembahasan, kita menggunakan sistem koordinat dimana $(0,0)$ berada di kiri atas, dan untuk titik (c, d), $c > 0$ dan $d > 0$, titik tersebut berada di kanan bawah.

Mula-mula, mari kita coba mengerjakan versi lebih mudah dari soal ini. Anggap bahwa hanya ada 2 orang, yaitu A dan B. Misal, A berada di (X_1, Y_1), dan titik B nantinya berada di (X_2, Y_2). Maka, titik mana saja yang bisa menjadi lokasi B? Ya, titik-titik yang berada di daerah persegi panjang dengan pojok kiri atas $(A_x - X_1, A_y - Y_1)$ dan pojok kanan bawah $(B_x - X_1, B_y - Y_1)$. Bagaimana cara menghitungnya jika ada banyak kemungkinan titik B? Kita bisa memanfaatkan struktur data yang bisa melakukan *range sum query* pada 2 dimensi. Contohnya, *segment tree* $2D$ atau *fenwick tree* $2D$. Tentunya, kita juga harus meng-*compress* koordinat yang ada, agar *memory* yang dibutuhkan tidak besar. Bagaimana caranya? Kita cukup mencatat nilai x dan y yang pernah muncul pada input, dan akan dimasukkan dalam struktur data, dalam hal ini semua titik (X_2, Y_2).

Nah, permasalahannya, saat ini kita memiliki 4 orang. Bagaimana cara mengakalinya? Kita bisa menggunakan pendekatan *meet in the middle*. Caranya, kita "menggabungkan" tiap 2 orang, sebut saja 2 orang pertama dan 2 orang terakhir. Tentunya, masing-masing penggabungan menghasilkan N^2 kemungkinan. Setelah penggabungan tersebut, kita kembali pada versi mudah yang kita bahas pada paragraf sebelumnya, sehingga kita bisa menggunakan cara yang sama untuk menyelesaikannya.

Total kompleksitas yang diharapkan adalah $O(N^2 \log^2 N)$.

F. Pusing Pusing Poligon

Tag

geometri

Pembahasan

Soal ini dapat diselesaikan dengan menggunakan sebuah observasi penting. Yaitu, apabila sebuah poligon reguler diputar dengan periode $N \cdot K$, maka irisan poligon reguler tersebut akan menjadi poligon reguler dengan jumlah sisi sebanyak $N \cdot K$. Ini dapat dibuktikan dengan melihat sebagian rotasi. Apabila poligon tersebut telah berputar selama K detik, maka poligon tersebut akan tepat bertumpukan sempurna dengan poligon awalnya, sehingga setiap sisi poligon akan terbagi menjadi K bagian yang sama panjang dan memiliki sudut yang sama. Karena poligon awal memiliki N sisi, maka jumlah sisi akhir adalah $N \cdot K$ sisi.

Untuk menghitung luas sebuah poligon reguler dapat menggunakan rumus yang dapat Anda cari sendiri dengan geometri dasar serta trigonometri. Kompleksitas akhir untuk solusi ini adalah $O(1)$.

G. Segmen Sirkular

Tag

matematika, counting

Pembahasan

Trik untuk mempermudah penghitungan adalah dengan memasangkan setiap entry pada deret B dengan entry pada deret A secara satu persatu. Perlu diketahui bahwa jumlah perkalian (penggeseran) yang diperlukan :

$$LCM(A,B) \ / \ B \ = A \ / \ GCD(A,B)$$

Dengan mengetahui jumlah penggeseran yang diperlukan maka kita dapat menyimpulkan bahwa setiap entry I pada deret B akan dipasangkan oleh semua entry J pada deret A dimana $J \ (mod \ GCD(A,B)) \ \equiv \ I \ (mod \ GCD(A,B))$.

Contoh untuk $B = 6$ dan $A = 9$:

- entry 1 dan 4 pada deret B akan dipasangkan dengan entry 1, 4, dan 7 pada deret A
- entry 2 dan 5 pada deret B akan dipasangkan dengan entry 2, 5, dan 8 pada deret A
- entry 3 dan 6 pada deret B akan dipasangkan dengan entry 3, 6, dan 9 pada deret A

Sehingga jumlahan hasil perkalian yang dilakukan oleh Pak Chanek adalah :

$$(1^2 + 4^2) \cdot (1^2 + 4^2 + 7^2) + (2^2 + 5^2) \cdot (2^2 + 5^2 + 8^2) + (3^2 + 6^2) \cdot (3^2 + 6^2 + 9^2)$$

Dengan mengetahui sifat tersebut, kita dapat membangun sebuah formula untuk menghitung jumlahan hasil perkalian tersebut. Misal, $F = GCD(A,B)$. Maka, formulanya adalah:

$$\sum_{j=1}^{F} (\sum_{i=0}^{\frac{A}{F}-1} (F \cdot i + j)^2 \cdot \sum_{i=0}^{\frac{B}{F}-1} (F \cdot i + j)^2)$$

Dengan menggunakan formula tersebut, kita bisa menyelesaikan soal ini dengan algoritma dengan kompleksitas $O(A + B)$.Untuk mempercepat perhitungan, mari kita bongkar sedikit formula tersebut.

$$\sum_{j=1}^{F} (\sum_{i=0}^{\frac{A}{F}-1} (F \cdot i + j)^2 \cdot \sum_{i=0}^{\frac{B}{F}-1} (F \cdot i + j)^2)$$

$$= \sum_{j=1}^{F} \left(\sum_{i=0}^{\frac{A}{F}-1} (F^2 i^2 + 2Fi + j^2) \cdot \sum_{i=0}^{\frac{B}{F}-1} (F^2 i^2 + 2Fi + j^2) \right)$$

Perlu diketahui bahwa :

$$X_0 = \sum_{i=0}^{N} i^0 = N + 1$$

$$Y_0 = \sum_{i=0}^{M} i^0 = M + 1$$

$$X_1 = \sum_{i=0}^{N} i^1 = \frac{N(N+1)}{2}$$

$$Y_1 = \sum_{i=0}^{M} i^1 = \frac{M(M+1)}{2}$$

$$X_2 = \sum_{i=0}^{N} i^2 = \frac{N(N+1)(2N+1)}{6}$$

$$Y_2 = \sum_{i=0}^{N} i^2 = \frac{M(M+1)(2M+1)}{6}$$

Dengan:

$$N = \frac{A}{F} - 1$$

$$M = \frac{B}{F} - 1$$

Sehingga:

$$\sum_{j=1}^{F} \left(\sum_{i=0}^{\frac{A}{F}-1} (F^2 i^2 + 2Fi + j^2) \cdot \sum_{i=0}^{\frac{B}{F}-1} (F^2 i^2 + 2Fi + j^2) \right)$$

$$= \sum_{j=1}^{F} ((F^2 X_2 + 2FjX_1 + j^2 X_0) \cdot (F^2 Y_2 + 2FjY_1 + j^2 Y_0))$$

Dengan menggunakan formula saat ini, kita bisa menyelesaikan soal ini dengan algoritma dengan kompleksitas $O(GCD(A, B))$. Untuk mempercepat perhitungan, mari kita bongkar sedikit lagi formula tersebut.

$$\sum_{j=1}^{F} ((F^2 X_2 + 2FjX_1 + j^2 X_0) \cdot (F^2 Y_2 + 2FjY_1 + j^2 Y_0))$$

$$= \sum_{j=1}^{F} (F^4 X_2 Y_2 + 4F^2 X_1 Y_1 j^2 + j^4 X_0 Y_0 + 2F^3 X_1 Y_2 j + 2F^3 X_2 Y_1 j)$$

$$+ \sum_{j=1}^{F} (F^2 X_0 Y_2 j^2 + F^2 X_2 Y_0 j^2 + 2F X_0 Y_1 j^3 + 2F X_1 Y_0 j^3)$$

Perlu diketahui (lagi) bahwa:

$$Z_1 = \sum_{i=0}^{F} i^1 = \frac{F(F+1)}{2}$$

$$Z_2 = \sum_{i=0}^{F} i^2 = \frac{F(F+1)(2F+1)}{6}$$

$$Z_3 = \sum_{i=0}^{F} i^3 = \left(\frac{F(F+1)}{2}\right)^2$$

$$Z_4 = \sum_{i=0}^{F} i^4 = \frac{6F^5 + 15F^4 + 10F^3 - F}{30}$$

Sehingga bisa kita dapatkan:

$$\sum_{j=1}^{F} (F^4 X_2 Y_2 + 4F^2 X_1 Y_1 j^2 + j^4 X_0 Y_0 + 2F^3 X_1 Y_2 j + 2F^3 X_2 Y_1 j)$$

$$+ \sum_{j=1}^{F} (F^2 X_0 Y_2 j^2 + F^2 X_2 Y_0 j^2 + 2F X_0 Y_1 j^3 + 2F X_1 Y_0 j^3)$$

$$= F^5 X_2 Y_2 + 4F^2 X_1 Y_1 Z_2 + Z_4 X_0 Y_0 + 2F^3 X_1 Y_2 Z_1 + 2F^3 X_2 Y_1 Z_1$$
$$+ F^2 X_0 Y_2 Z_2 + F^2 X_2 Y_0 Z_2 + 2F X_0 Y_1 Z_3 + 2F X_1 Y_0 Z_3$$

Dengan menggunakan formula akhir ini, kita bisa menyelesaikan soal ini dengan algoritma dengan kompleksitas $O(log(\min(A, B)))$ untuk mencari F, sehingga masalah selesai.

H. Tumbuh-Tumbuhan

Tag

dynamic programming

Pembahasan

Terdapat 2 bagian dalam penyelesaian soal ini, yaitu *brute force* dan *dynamic programming*. Untuk $N \leq 5$, tentunya kita bisa mencoba semua kemungkinan permutasi dalam $O(N \cdot N!)$. Permasalahannya, di sini N bisa mencapai 8.000. Ternyata, kita bisa menjalankan strategi *dynamic programming* untuk soal ini.

Perhatikan bahwa, untuk suatu tumbuhan ke-i, sebenarnya kita hanya ingin tahu, urutan tumbuhnya dibandingkan tumbuhan ke-$(i-2)$, $(i-1)$, $(i+1)$, dan $(i+2)$. Ini membawa kita ke suatu DP dengan *state (indeks, permutasi)*, dimana permutasi memiliki $5! = 120$ kemungkinan nilai berbeda. Misal, permutasi sekarang merepresentasikan urutan $(1, 3, 4, 5, 2)$. Ini berarti:

- Tumbuhan pada posisi $(indeks - 2)$ tumbuh $(4 - 1) = 3$ hari sebelum tumbuhan pada posisi indeks.
- Tumbuhan pada posisi $(indeks - 1)$ tumbuh $(4 - 3) = 1$ hari sebelum tumbuhan pada posisi indeks.
- Tumbuhan pada posisi $(indeks + 1)$ tumbuh $(5 - 4) = 1$ hari setelah tumbuhan pada posisi indeks.
- Tumbuhan pada posisi $(indeks + 2)$ tumbuh $(4 - 2) = 2$ hari sebelum tumbuhan pada posisi indeks.

Dari sini, kita bisa menyimpulkan bahwa saat tumbuhan ke-indeks tumbuh, sudah ada 3 tumbuhan lain yang tumbuh, pada jarak ≤ 2 dari indeks.

Mengingat permutasi ini menyatakan urutan relatif, maka transisi ke indeks setelahnya dapat dilakukan dengan menghapus urutan tumbuhan $indeks - 2$, merombak urutan relatif, melakukan *brute force* untuk menentukan posisi relatif tumbuhan $indeks + 3$, menentukan permutasi yang menyatakan urutan relatif baru, dan berpindah ke *state (indeks+1, permutasi')*.

Mengingat terdapat sedikit kasus khusus, kita bisa melakukan *brute force* untuk menentukan urutan relatif 5 tumbuhan pertama, lalu menjalankan DP di atas. Maka, kompleksitas akhir untuk solusi ini adalah $O(N \cdot K \cdot K!)$, dengan $K = 5$.