Convex Hull DP Optimization

Firman Hadi P. – TKTPL Genap 2018/2019

Fakultas Ilmu Komputer, Universitas Indonesia

28 Februari 2019

Outline

1. *Motivasi*
2. *Query* pada kumpulan persamaan garis
3. Aplikasi pada DP

Motivasi

SPOJ GOODG - Good Inflation

Anda memegang sebuah balon selama N $(1 \leq N \leq 10^6)$ menit.

Pada menit 0, ukuran balon mempunyai volume 0.
Terdapat N buah tawaran, tawaran ke-i akan:

- Menambah volume balon sebesar a_i $(1 \leq a_i \leq 10^6)$ pada menit ke-i.
- Namun seterusnya volume balon akan berkurang sebanyak d_i $(1 \leq d_i \leq 10^6)$ per menit sampai Anda mengambil tawaran berikutnya.

Berapa volume terbesar balon yang bisa dicapai pada menit $N + 1$?

Motivasi

Permasalahan sebelumnya (tentu saja) dapat diselesaikan dengan DP.

Motivasi

Permasalahan sebelumnya (tentu saja) dapat diselesaikan dengan DP.

Definisikan $V(x)$ sebagai volume terbesar yang bisa dicapai pada menit ke-x dan kita ingin mengambil tawaran ke-x.

$$V(x) = \begin{cases} 0 & \text{jika } x = 0 \\ a_x + \max\limits_{i<x}\{V(i) - (x-i)d_i\} & \text{jika } x \neq 0 \end{cases}$$

Tambahkan tawaran *dummy*: 0 dan $N+1$ dengan masing-masing $a_i = 0$ dan $d_i = 0$.

Motivasi

Jawaban terdapat pada $V(N+1)$.

Kompleksitas memori: $O(N)$
Kompleksitas waktu: $O(N^2)$?

Motivasi

Jawaban terdapat pada $V(N+1)$.

Kompleksitas memori: $O(N)$
Kompleksitas waktu: $O(N^2)$? TLE!

Bagaimana cara mempercepat perhitungan tersebut?

Untuk sejenak, lupakan permasalahan sebelumnya. Mari kita beralih ke permasalahan yang lebih *"simple"*.

Outline

Permasalahan yang lebih *"simple"*

Query pada kumpulan persamaan garis

Diberikan N buah persamaan garis $y_i = f_i(x) = m_i x + c_i$ dan Q buah *query* dengan x_{q_i}:

Berapa nilai ekstrim (dalam hal ini, maksimum) dari setiap persamaan garis pada $x = x_{q_i}$?

Permasalahan yang lebih *"simple"*

Query pada kumpulan persamaan garis

Diberikan N buah persamaan garis $y_i = f_i(x) = m_i x + c_i$ dan Q buah *query* dengan x_{q_i}:

Berapa nilai ekstrim (dalam hal ini, maksimum) dari setiap persamaan garis pada $x = x_{q_i}$?

Pendekatan *Brute-force*: Untuk setiap *query*, masukkan x_{q_i} untuk setiap persamaan garis $\rightarrow O(NQ)$.

Permasalahan yang lebih *"simple"*

Query pada kumpulan persamaan garis

Diberikan N buah persamaan garis $y_i = f_i(x) = m_i x + c_i$ dan Q buah *query* dengan x_{q_i}:

Berapa nilai ekstrim (dalam hal ini, maksimum) dari setiap persamaan garis pada $x = x_{q_i}$?

Pendekatan *Brute-force*: Untuk setiap *query*, masukkan x_{q_i} untuk setiap persamaan garis $\rightarrow O(NQ)$.

Bisa dipercepat menjadi $O(\log N)$ untuk setiap *query* menggunakan *binary search* $\rightarrow O(Q \log N)$.

Permasalahan yang lebih *"simple"*

Query pada kumpulan persamaan garis

Diberikan N buah persamaan garis $y_i = f_i(x) = m_i x + c_i$ dan Q buah *query* dengan x_{q_i}:

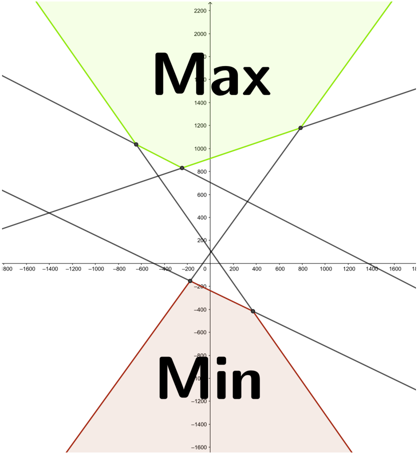
Berapa nilai ekstrim (dalam hal ini, maksimum) dari setiap persamaan garis pada $x = x_{q_i}$?

Pendekatan *Brute-force*: Untuk setiap *query*, masukkan x_{q_i} untuk setiap persamaan garis $\rightarrow O(NQ)$.

Bisa dipercepat menjadi $O(\log N)$ untuk setiap *query* menggunakan *binary search* $\rightarrow O(Q \log N)$. *How?*

Mempercepat *query*

Perhatikan contoh untuk beberapa garis:



Mempercepat *query*

Observasi

- Beberapa garis (tidak semua) akan memberikan nilai maksimum untuk suatu rentang x.

Mempercepat *query*

Observasi

- Beberapa garis (tidak semua) akan memberikan nilai maksimum untuk suatu rentang x.
- Untuk setiap x dari kiri ke kanan ($-\infty \rightarrow \infty$), urutan garis yang menjadi maksimum mempunyai gradien yang menaik!

Mempercepat *query*

Observasi

- Beberapa garis (tidak semua) akan memberikan nilai maksimum untuk suatu rentang x.
- Untuk setiap x dari kiri ke kanan $(-\infty \to \infty)$, urutan garis yang menjadi maksimum mempunyai gradien yang menaik!

Bagaimana cara memanfaatkan properti di atas?

Mempercepat *query*

Misalkan kita sudah menyimpan setiap garis y_i yang memberikan nilai maksimum pada x pada suatu rentang $[l_i, r_i]$ (abaikan garis yang tidak menjadi maksimum untuk rentang manapun). Definisikan garis-garis tersebut sebagai garis relevan (y_{rel}).

Mempercepat *query*

Misalkan kita sudah menyimpan setiap garis y_i yang memberikan nilai maksimum pada x pada suatu rentang $[l_i, r_i]$ (abaikan garis yang tidak menjadi maksimum untuk rentang manapun). Definisikan garis-garis tersebut sebagai garis relevan (y_{rel}).

Urutkan garis-garis relevan tersebut berdasarkan gradien menaik.

Mempercepat *query*

Misalkan kita sudah menyimpan setiap garis y_i yang memberikan nilai maksimum pada x pada suatu rentang $[l_i, r_i]$ (abaikan garis yang tidak menjadi maksimum untuk rentang manapun). Definisikan garis-garis tersebut sebagai garis relevan (y_{rel}).

Urutkan garis-garis relevan tersebut berdasarkan gradien menaik.

Perhatikan bahwa: Rentang x untuk suatu garis terbatas pada titik potong garis tersebut dengan garis di sebelah kirinya dan kanannya.

Mempercepat *query*

Misalkan kita sudah menyimpan setiap garis y_i yang memberikan nilai maksimum pada x pada suatu rentang $[l_i, r_i]$ (abaikan garis yang tidak menjadi maksimum untuk rentang manapun). Definisikan garis-garis tersebut sebagai garis relevan (y_{rel}).

Urutkan garis-garis relevan tersebut berdasarkan gradien menaik.

Perhatikan bahwa: Rentang x untuk suatu garis terbatas pada titik potong garis tersebut dengan garis di sebelah kirinya dan kanannya.

Dengan kata lain, misalkan p adalah titik potong pada garis $y_{rel_{i-1}}$ dan y_{rel_i}, dan q adalah titik potong pada garis y_{rel_i} dan $y_{rel_{i+1}}$, nilai x sehingga y_{rel_i} memberikan nilai maksimum berada pada rentang $[p_x, q_x]$.

Mempercepat *query*

Kasus khusus:

- Pada garis pertama, batas kirinya adalah $-\infty$, dan
- Pada garis terakhir, batas kanannya adalah ∞.

Mempercepat *query*

Kasus khusus:

- Pada garis pertama, batas kirinya adalah $-\infty$, dan
- Pada garis terakhir, batas kanannya adalah ∞.

Kita dapat melakukan *binary search* pada kumpulan garis-garis relevan untuk mendapatkan nilai maksimum.

Mempercepat *query*

Algorithm LineQuery, time complexity: $O(\log N)$

Input: x and relevant lines sorted by ascending slope y_{rel}

Output: Maximum value for every line on x.

$L \leftarrow 1$
$R \leftarrow |y_{rel}|$ $\triangleright$ $|\cdot|$ = cardinality operator
while $L \le R$ **do**
    $MID \leftarrow \left\lfloor \frac{L+R}{2} \right\rfloor$
    if x lies on the right of ${y_{rel}}_{MID}$'s interval **then**
        $L \leftarrow MID + 1$
    else if x lies on the left of ${y_{rel}}_{MID}$'s interval **then**
        $R \leftarrow MID - 1$
    else
        return value of ${y_{rel}}_{MID}$ on x
    end if
end while

Belum selesai...

Bagaimana caranya mengetahui relevan tidaknya suatu garis?

Mengumpulkan garis-garis relevan akan lebih mudah apabila kita sebelumnya mengetahui semua garis.

Belum selesai...

Bagaimana caranya mengetahui relevan tidaknya suatu garis?

Mengumpulkan garis-garis relevan akan lebih mudah apabila kita sebelumnya mengetahui semua garis.

Garis-garis tersebut akan diproses satu per satu dimulai dari gradien paling kecil.

Mencari garis relevan

Lemma 1.1

Jika terdapat dua buah garis $y_1 = f_1(x) = m_1x + c_1$ *dan* $y_2 = f_2(x) = m_2x + c_2$ *dengan* $m_1 < m_2$, *berlaku:*

$f_1(x) \geq f_2(x)$ *pada* $x \leq x_{12}$, *dan*
$f_1(x) \leq f_2(x)$ *pada* $x \geq x_{12}$
yang mana x_{ij} *adalah komponen* x *pada titik potong* y_i *dan* y_j.

Mencari garis relevan

Lemma 1.1

Jika terdapat dua buah garis $y_1 = f_1(x) = m_1x + c_1$ *dan* $y_2 = f_2(x) = m_2x + c_2$ *dengan* $m_1 < m_2$*, berlaku:*

$f_1(x) \geq f_2(x)$ *pada* $x \leq x_{12}$*, dan*
$f_1(x) \leq f_2(x)$ *pada* $x \geq x_{12}$
yang mana x_{ij} *adalah komponen* x *pada titik potong* y_i *dan* y_j*.*

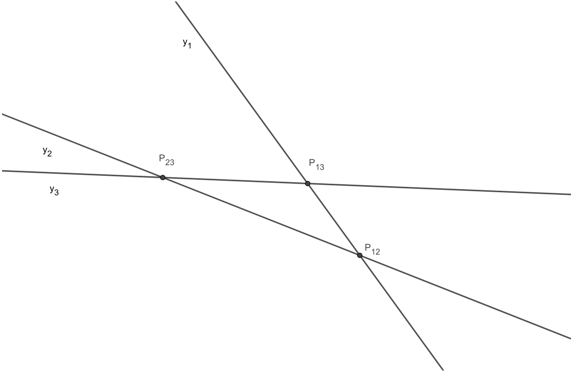
Bukti

Mudah dibuktikan melalui representasi geometri. Dapat juga dibuktikan menggunakan aljabar.

Mencari garis relevan

Lemma 1.2

Jika terdapat tiga buah garis y_1, y_2, y_3 *dengan* $m_1 < m_2 < m_3$ *dan* $x_{13} \leq x_{12}$, *maka* $x_{23} \leq x_{12}$.



$$P_{ij} = (x_{ij}, f_i(x_{ij}))$$

Mencari garis relevan

Teorema 1.1

Jika terdapat tiga buah garis y_1, y_2, y_3 *dengan* $m_1 < m_2 < m_3$ *dan* $x_{13} \leq x_{12}$, *maka* y_2 *bukanlah garis relevan.*

Mencari garis relevan

Teorema 1.1

Jika terdapat tiga buah garis y_1, y_2, y_3 *dengan* $m_1 < m_2 < m_3$ *dan* $x_{13} \leq x_{12}$*, maka* y_2 *bukanlah garis relevan.*

Bukti

Melalui *Lemma 1.2*, didapat juga $x_{23} \leq x_{12}$.
Melalui *Lemma 1.1* juga didapat bahwa $f_3(x) \geq f_1(x)$ pada $x \geq x_{13}$ dan $f_3(x) \geq f_2(x)$ pada $x \geq x_{23}$.

Dikarenakan $x_{23} \leq x_{12}$, maka garis y_2 tidak akan memberikan nilai maksimum untuk x manapun karena sudah 'didahului' oleh garis y_3 sebelum peralihan maksimum dari y_1 ke y_2. (*perhatikan gambar pada Lemma 1.2*)

Mencari garis relevan

Teorema 1.1 sudah cukup untuk mengumpulkan seluruh garis-garis relevan.

Mencari garis relevan

Teorema 1.1 sudah cukup untuk mengumpulkan seluruh garis-garis relevan.

Urutkan seluruh garis berdasarkan gradien secara menaik.

Mencari garis relevan

Teorema 1.1 sudah cukup untuk mengumpulkan seluruh garis-garis relevan.

Urutkan seluruh garis berdasarkan gradien secara menaik.
Masukkan garis pertama ke dalam *stack*.

Mencari garis relevan

Teorema 1.1 sudah cukup untuk mengumpulkan seluruh garis-garis relevan.

Urutkan seluruh garis berdasarkan gradien secara menaik.
Masukkan garis pertama ke dalam *stack*.
Masukkan garis sisanya satu per satu, dengan kondisi apabila *stack* berisi lebih dari 1 garis:

- Misalkan y_1 adalah garis kedua dari atas *stack*, y_2 adalah garis paling atas di *stack*, dan y_3 adalah garis yang sedang ingin dimasukkan dalam *stack*,
- Apabila $x_{13} \leq x_{12}$, keluarkan y_2 dari *stack*.
- Ulangi prosedur tersebut sampai kondisi di atas tidak terpenuhi.

Mencari garis relevan

Teorema 1.1 sudah cukup untuk mengumpulkan seluruh garis-garis relevan.

Urutkan seluruh garis berdasarkan gradien secara menaik.
Masukkan garis pertama ke dalam *stack*.
Masukkan garis sisanya satu per satu, dengan kondisi apabila *stack* berisi lebih dari 1 garis:

- Misalkan y_1 adalah garis kedua dari atas *stack*, y_2 adalah garis paling atas di *stack*, dan y_3 adalah garis yang sedang ingin dimasukkan dalam *stack*,
- Apabila $x_{13} \leq x_{12}$, keluarkan y_2 dari *stack*.
- Ulangi prosedur tersebut sampai kondisi di atas tidak terpenuhi.

Garis-garis yang masih ada di dalam *stack* merupakan garis-garis yang relevan.

Mencari garis relevan

Algorithm FindRelevantLines, time complexity: $O(N)$

Input: y = array of lines with size N
Output: y_{rel} = relevant lines, sorted by increasing gradient

Sort y by increasing gradient
Push y_1 onto stack
for each y_i in y, $i \in \{2, 3, ...N\}$ **do**
    while stack size is greater than 1 **do**
        $l_1, l_2 \leftarrow$ second, first line from top of the stack
        $l_3 \leftarrow y_i$
        $x_{12}, x_{13} \leftarrow$ x-intersection of l_1 and l_2, l_1 and l_3
        if $x_{13} \leq x_{12}$ **then** pop l_2 from stack **else break**
    end while
    Push y_i onto stack
end for
return y_{rel} = stack contents (bottom to top)

Mencari garis relevan

Algoritma sebelummnya menggunakan asumsi bahwa kita sudah mengetahui semua garis yang akan dimasukkan (*offline*) sebelum melakukan semua *query*.

Bagaimana jika kita tidak mengetahuinya (*online*)?

Mencari garis relevan

Algoritma sebelummnya menggunakan asumsi bahwa kita sudah mengetahui semua garis yang akan dimasukkan (*offline*) sebelum melakukan semua *query*.

Bagaimana jika kita tidak mengetahuinya (*online*)? Kita bisa saja:

- Berharap (atau 'memaksa') garis-garis yang kita masukkan ke *stack* satu per satu sudah pasti terurut berdasarkan gradien.
- Jika tidak memungkinkan, gunakan struktur data `set` (C++) atau *Li Chao Tree*.

Outline

Kembali ke masalah sebelumnya...

Solusi masalah sebelumnya

Definisikan $V(i)$ sebagai volume terbesar yang bisa dicapai pada menit ke-i dan kita ingin mengambil tawaran ke-i.

$$V(i) = \begin{cases} 0 & \text{jika } i = 0 \\ a_i + \max_{j<i}\{V(j) - (i-j)d_j\} & \text{jika } i \neq 0 \end{cases}$$

Tambahkan tawaran *dummy*: 0 dan $N+1$ dengan masing-masing $a_i = 0$ dan $d_i = 0$. Solusi terdapat pada $V(N+1)$.

Apakah kita bisa mengubah permasalahan tersebut menjadi *query* pada kumpulan persamaan garis?

Kembali ke masalah sebelumnya...

Menjabarkan $\max_{j<i}\{V(j) - (i-j)d_j\}$ menghasilkan

$$\begin{aligned}\max_{j<i}\{V(j) - (i-j)d_j\} &= \max_{j<i}\{V(j) - i\,d_j + j\,d_j\} \\ &= \max_{j<i}\{-d_j\,i + j\,d_j + V(j)\}\end{aligned}$$

Kembali ke masalah sebelumnya...

Menjabarkan $\max_{j<i}\{V(j) - (i-j)d_j\}$ menghasilkan

$$\begin{aligned}\max_{j<i}\{V(j) - (i-j)d_j\} &= \max_{j<i}\{V(j) - i\,d_j + j\,d_j\} \\ &= \max_{j<i}\{-d_j\,i + j\,d_j + V(j)\}\end{aligned}$$

Misalkan $m_j = -d_j$, $x_i = i$ dan $c_j = j\,d_j + V(j)$, maka

$$\max_{j<i}\{V(j) - (i-j)d_j\} = \max_{j<i}\{m_j x_i + c_j\}$$

Kembali ke masalah sebelumnya...

Menjabarkan $\max_{j<i}\{V(j) - (i-j)d_j\}$ menghasilkan

$$\begin{aligned}\max_{j<i}\{V(j) - (i-j)d_j\} &= \max_{j<i}\{V(j) - i\,d_j + j\,d_j\} \\ &= \max_{j<i}\{-d_j\,i + j\,d_j + V(j)\}\end{aligned}$$

Misalkan $m_j = -d_j$, $x_i = i$ dan $c_j = j\,d_j + V(j)$, maka

$$\max_{j<i}\{V(j) - (i-j)d_j\} = \max_{j<i}\{m_j x_i + c_j\}$$

Persamaan garis!

Belum selesai...

Perhitungan DP tersebut dapat dilakukan secara *bottom up* sambil menambahkan persamaan garis satu per satu.

Belum selesai...

Perhitungan DP tersebut dapat dilakukan secara *bottom up* sambil menambahkan persamaan garis satu per satu.

Namun, terlihat bahwa $m_i = -d_i$. Gradien dari garis yang akan kita masukkan tidak selalu terurut secara menaik sehingga kita tidak bisa menggunakan struktur data *stack*.

Belum selesai...

Perhitungan DP tersebut dapat dilakukan secara *bottom up* sambil menambahkan persamaan garis satu per satu.

Namun, terlihat bahwa $m_i = -d_i$. Gradien dari garis yang akan kita masukkan tidak selalu terurut secara menaik sehingga kita tidak bisa menggunakan struktur data *stack*.

Bisakah kita 'memaksa' agar gradiennya terurut?

Belum selesai...

Perhitungan DP tersebut dapat dilakukan secara *bottom up* sambil menambahkan persamaan garis satu per satu.

Namun, terlihat bahwa $m_i = -d_i$. Gradien dari garis yang akan kita masukkan tidak selalu terurut secara menaik sehingga kita tidak bisa menggunakan struktur data *stack*.

Bisakah kita 'memaksa' agar gradiennya terurut? Bisa!

Merombak ulang solusi

Formulasi ulang DP

Definisikan $V'(i)$ sebagai volume maksimum yang bisa kita dapat pada menit ke-$N+1$ apabila pada awalnya kita mengambil tawaran ke-i.

$$V'(i) = \begin{cases} 0 & \text{jika } i = N+1 \\ a_i + \max_{i<j\leq N+1} \{V'(j) - (j-i)d_i\} & \text{jika } i < N+1 \end{cases}$$

Solusi terdapat pada $\max\{V'(1), V'(2), ..., V'(N+1)\}$.

Merombak ulang solusi

Menjabarkan $\max_{i<j\leq N+1}\{V'(j)-(j-i)d_i\}$ menghasilkan

$$\begin{aligned}\max_{i<j\leq N+1}\{V'(j)-(j-i)d_i\} &= \max_{i<j\leq N+1}\{V'(j)-j\,d_i+i\,d_i\}\\ &= \max_{i<j\leq N+1}\{V'(j)-j\,d_i\}+i\,d_i\\ &= \max_{i<j\leq N+1}\{-j\,d_i+V'(j)\}+i\,d_i\end{aligned}$$

Merombak ulang solusi

Menjabarkan $\max\limits_{i<j\leq N+1} \{V'(j) - (j-i)d_i\}$ menghasilkan

$$\begin{aligned}\max_{i<j\leq N+1} \{V'(j) - (j-i)d_i\} &= \max_{i<j\leq N+1} \{V'(j) - j\,d_i + i\,d_i\} \\ &= \max_{i<j\leq N+1} \{V'(j) - j\,d_i\} + i\,d_i \\ &= \max_{i<j\leq N+1} \{-j\,d_i + V'(j)\} + i\,d_i\end{aligned}$$

Misalkan $m_j = -j$, $x_i = d_i$, dan $c_j = V'(j)$, maka

$$\max_{i<j\leq N+1} \{V'(j) - (j-i)d_i\} = \max_{i<j\leq N+1} \{m_j x_i + c_j\} + i\,d_i$$

Merombak ulang solusi

Menjabarkan $\max\limits_{i<j\leq N+1}\{V'(j)-(j-i)d_i\}$ menghasilkan

$$\begin{aligned}\max_{i<j\leq N+1}\{V'(j)-(j-i)d_i\} &= \max_{i<j\leq N+1}\{V'(j)-j\,d_i+i\,d_i\}\\ &= \max_{i<j\leq N+1}\{V'(j)-j\,d_i\}+i\,d_i\\ &= \max_{i<j\leq N+1}\{-j\,d_i+V'(j)\}+i\,d_i\end{aligned}$$

Misalkan $m_j=-j$, $x_i=d_i$, dan $c_j=V'(j)$, maka

$$\max_{i<j\leq N+1}\{V'(j)-(j-i)d_i\} = \max_{i<j\leq N+1}\{m_jx_i+c_j\}+i\,d_i$$

Persamaan garis!

Merombak ulang solusi

Apa bedanya?

Perhitungan $V(i)$ akan dimulai dari 0 sampai $N + 1$, sedangkan $V'(i)$ akan dimulai dari $N + 1$ sampai 1.

Merombak ulang solusi

Apa bedanya?

Perhitungan $V(i)$ akan dimulai dari 0 sampai $N+1$, sedangkan $V'(i)$ akan dimulai dari $N+1$ sampai 1.

Perbedaan paling penting adalah gradien garis-garis pada perhitungan $V'(i)$ **akan terurut secara menaik**.

Merombak ulang solusi

Apa bedanya?

Perhitungan $V(i)$ akan dimulai dari 0 sampai $N+1$, sedangkan $V'(i)$ akan dimulai dari $N+1$ sampai 1.

Perbedaan paling penting adalah gradien garis-garis pada perhitungan $V'(i)$ **akan terurut secara menaik**.

Sekarang, kita dapat menggunakan struktur data *stack* untuk memasukkan garis.

Solusi

Algorithm GOODG, time complexity: $O(N \log N)$

Input: N, N offers (a_i, d_i)
Output: Maximum possible volume of balloon at (N+1) minutes

```
ans ← 0, y_rel ← empty stack, V[N] ← zero array
Push line y_{N+1} = −(N + 1)x + 0 onto y_rel
for i in [N, N − 1, ..., 2, 1] do
    maxval ← result of LINEQUERY on y_rel with x = d_i
    V[i] = a_i + maxval + i d_i
    ans ← max{ans, V[i]}
    Push line y_i = −ix + V[i] onto y_rel, also remove irrelevant lines
end for
return ans
```

Yay!

ID	DATE	USER	PROBLEM	RESULT	TIME	MEM	LANG
23244203	2019-02-16 17:26:06	Firman Hadi P.	Good Inflation	**accepted** edit ideone it	0.40	39M	CPP14

Kesimpulan

Beberapa transisi DP dapat dipercepat menggunakan analogi persamaan garis.

Kesimpulan

Beberapa transisi DP dapat dipercepat menggunakan analogi persamaan garis.

Beberapa solusi DP yang membutuhkan kompleksitas $O(N^2)$ dapat dipercepat menjadi $O(N \log N)$ atau bahkan $O(N)$ apabila *query* menggunakan nilai x yang monoton sehingga tidak diperlukan *binary search*.

Kesimpulan

Beberapa transisi DP dapat dipercepat menggunakan analogi persamaan garis.

Beberapa solusi DP yang membutuhkan kompleksitas $O(N^2)$ dapat dipercepat menjadi $O(N \log N)$ atau bahkan $O(N)$ apabila *query* menggunakan nilai x yang monoton sehingga tidak diperlukan *binary search*.

Anda bisa menggunakan `vector` pada C++ sebagai *stack* untuk mempermudah.

Kesimpulan

Beberapa transisi DP dapat dipercepat menggunakan analogi persamaan garis.

Beberapa solusi DP yang membutuhkan kompleksitas $O(N^2)$ dapat dipercepat menjadi $O(N \log N)$ atau bahkan $O(N)$ apabila *query* menggunakan nilai x yang monoton sehingga tidak diperlukan *binary search*.

Anda bisa menggunakan `vector` pada C++ sebagai *stack* untuk mempermudah.

Dengan sedikit modifikasi, Anda dapat juga menggunakan teknik ini untuk mencari nilai minimum.

Referensi

- cp-algorithms.com - Convex Hull Trick
- WCIPEG Wiki - Convex Hull Trick
- Codeforces Blog - Dynamic Programming Optimizations