CHUYÊN MỤC

- Cấu trúc dữ liệu
  - BIT
  - Deque
  - Hashing
  - Heap
  - IT
  - Queue
  - Set
  - Stack
- Cloud Platform
  - Server
- Khác
- Kiếm tiền Online
- Lập trình Mobile
  - Lập trình Android
- Lập trình Web
  - CSS
  - HTML
- MongoDB
- Ngôn ngữ lập trình
  - .NET
  - C++
  - Go
  - Java
  - MySQL
  - NodeJS
  - Pascal
  - PHP
- OJ
  - Codeforces
  - Kattis
  - PTIT
  - SPOJ
- Phương pháp
  - Đệ quy có nhớ
  - Nén số (rời rạc hóa mảng)
  - Nhân ma trận
  - Quy hoạch động
  - Quy hoạch động trạng thái
- Thủ thuật máy tính
- Thư viện
  - Đề thi
  - Giải thuật
  - Tài liệu
- Thuật toán
  - Bitmask
  - Duyệt phân tập
  - Đệ quy
  - Đồ thị
    - BFS
    - Cặp ghép

Một số kỹ thuật tối ưu hóa thuật toán quy hoạch động

10/08/2017 BY AIDA NANA LEAVE A COMMENT

LỜI NÓI ĐẦU

Quy hoạch động (QHĐ) là một lớp thuật toán rất quan trọng và có nhiều ứng dụng trong ngành khoa học máy tính. Trong các cuộc thi Olympic tin học hiện đại, QHĐ luôn là một trong những chủ đề chính. Tuy vậy, theo tôi thấy, tài liệu nâng cao về QHĐ bằng tiếng Việt hiện còn cực kỳ khan hiếm, dẫn đến học sinh/sinh viên Việt Nam bị hạn chế khả năng tiếp cận với những kỹ thuật hiện đại. Trong bài viết này, tôi sẽ trình bày một vài kỹ thuật để tối ưu hóa độ phức tạp của một số thuật toán QHĐ.

Lê Anh Đức A2K42-PBC

1. **Đổi biến**

Nhiều khi trong trạng thái QHĐ có một thành phần nào đấy với khoảng giá trị quá lớn, trong khi kết quả của hàm lại có khoảng giá trị nhỏ. Trong một vài trường hợp, ta có thể đảo nhãn để giảm số trạng thái.

Bài tập ví dụ

Longest Common Subsequence (bài toán cổ điển LCS)

(bộ test đi kèm)

Cho xâu A độ dài m, xâu B độ dài n. Hãy tìm độ dài xâu con chung dài nhất của hai xâu, chú ý là xâu con chung có thể không liên tiếp.

Giới hạn:

- **m <= 1 000 000**
- n <= 5 000
- Các kí tự trong cả hai xâu là các chữ cái tiếng Anh in hoa 'A'..'Z'

Input:

- DFS
- Dijkstra
- Floyd
- Kruskal
- LCA
- Luồng
- Prim
- Tree
- Hình học
  - Convex Hull
- Khác
- Tham lam
- Tìm kiếm nhị phân
- Toán học
  - Sàng nguyên tố
- Vét cạn
- Xử lý số lớn
- Toán rời rạc
- Trí tuệ nhân tạo
  - Học máy

BÀI VIẾT MỚI

- Giải thuật tìm kiếm Fibonacci
- Heroku là gì?
- GSON là gì?

- **Dòng thứ nhất chứa xâu A**
- Dòng thứ hai chứa xâu B

Output:

- **Dòng duy nhất chứa kết quả**

Ví dụ:

Input	Output
ADBCC ABCD	3

Lời giải

Thuật toán đơn giản

Gọi F(i, j) là LCS của hai tiền tố A[1..i] và B[1..j].

Khi đó ta có thể maximize F(i, j) theo F(i-1, j) và F(i, j-1).

Nếu A[i] = B[j] thì ta có thể cập nhật F(i, j) theo F(i-1, j-1) + 1.

Kết quả bài toán là F(m, n).

Độ phức tạp của thuật toán này là O(m*n), không khả thi với giới hạn của đề bài.

biến

_ = min(m, n);

rằng trong hàm QHĐ trên, các giá trị của F(i, j) sẽ không vượt quá L, trong khi đó ı thứ hai của trạng thái có thể khá lớn (lên tới MAXM = 1 000 000).

ối ưu hóa, ta sẽ đổi biến. Gọi dp(i, j) là vị trí k nhỏ nhất sao cho LCS(A[1..i], B[1..k])

nh các giá trị của dp(), ta sẽ QHĐ theo kiểu cập nhật đi, thay vì đi tìm công thức tiếp cho các dp(i, j).

Gọi nextPos(i, c) = j > i nhỏ nhất mà a[j] = c (với c là một ký tự từ 'A' đến 'Z').

Mảng nextPos[][] có thể tính trong O(M*26).

Như vậy ta có thể tính các giá trị QHĐ như sau:

- **Ban đầu khởi tạo các giá trị dp(i, j) = vô cùng, dp(0, 0) = 0.**
- For i và j tăng dần, với mỗi giá trị dp(i, j) khác vô cùng:

- Cập nhật dp(i+1, j) theo dp(i, j).
- Gọi k là vị trí xuất hiện tiếp theo của b[i+1] trong xâu A bắt đầu từ vị trí dp(i, j), tức là k = nextPos(dp(i, j), b[i+1]).
- Nếu tồn tại k, cập nhật dp(i+1, j+1) theo k.

Để tính kết quả, ta sẽ chỉ cần tìm j lớn nhất mà tồn tại dp(i, j) khác vô cùng.

```
#include <bits/stdc++.h>
```

- Cài đặt NodeJS và một số lỗi thường gặp
- Giới thiệu về MongoDB
- COMNET – spoj
- Tổng hợp tài liêu, đề thi môn Cơ Nhiệt
- LUBENICA – spoj

BÌNH LUẬN MỚI

- hieu4 trong SEQ198 – spoj
- Nguyễn Ngọc Trung trong Top 5 trường đại học tốt nhất để học công nghệ thông tin
- Phạm Văn Khánh trong SEQ198 – spoj
- Hoàng trong PBCDEM – SPOJ
- võ long trong LINEGAME – SPOJ
- võ long trong LINEGAME – SPOJ
- INFORMAC – spoj trong Tổng hợp tài liệu về thuật toán cặp ghép

QUẢN LÝ BLOG

- Đăng kí
- Đăng nhập
- RSS cho bài viết
- Dòng thông tin các phản hồi.
- WordPress.org


```
using namespace std;

const int M = 1e6 + 6;
const int N = 5005;

int dp[N][N];

char a[M], b[N];
int nextPos[M][26];
int m, n;

void minimize(int &a, int b) {
if (a == -1 || a > b) a = b;
}

int main() {
cin >> a + 1 >> b + 1;
m = strlen(a + 1); n = strlen(b + 1);
for (int c = 0; c < 26; ++c)
for (int i = m – 1; i >= 0; –i)
nextPos[i][c] = (a[i + 1] – 'A' == c) ? i + 1 :                nextPos[i + 1][c];
int maxLength = min(m, n);
memset(dp, -1, sizeof dp);
dp[0][0] = 0;
for (int i = 0; i < n; ++i) {
for (int j = 0; j <= i; ++j) if (dp[i][j] >= 0) {
minimize(dp[i + 1][j], dp[i][j]);
int new_value = nextPos[dp[i][j]][b[i + 1] – 'A'];
if (new_value > 0)
minimize(dp[i + 1][j + 1], new_value);
}
}
int ans = 0;
for (int j = maxLength; j > 0; –j) {
for (int i = j; i <= n; ++i)
if (dp[i][j] >= 0) ans = j;
if (ans != 0) break;
}
cout << ans << endl;
return 0;
}
```

Bài tập ví dụ

COMPUTER (VNOI Marathon 2010)

Submit: vn.spoj.com/problems/COMPUTER/

Công ty phần mềm XYZ mới mua x máy tính để bàn và y máy tính xách tay. Giá một máy tính để bàn là a đồng còn giá một máy tính xách tay là b đồng. Để tránh sự thắc mắc giữa các phòng bàn, Tổng giám đốc đã đưa ra cách phân bố các máy tính này về n phòng ban như sau:

- **Sắp xếp n phòng ban theo thứ tự về mức độ quan trọng của các phòng ban.**
- Tiến hành phân bố các máy tính cho các phòng ban bảo đảm nếu phòng ban i có mức độ quan trọng nhỏ hơn mức độ quan trọng của phòng ban j thì tổng giá trị máy tính được phân bố cho phòng ban i không được vượt quá tổng giá trị máy tính được phân bố cho phòng ban j.
- Phòng ban nhận được tổng giá trị máy tính nhỏ nhất là lớn nhất.

Là một lập trình viên giỏi nhưng lại thuộc phòng ban có mức độ quan trọng nhỏ nhất, Thắng muốn chứng tỏ tay nghề của mình với đồng nghiệp nên đã lập trình tính ra ngay được tổng giá trị máy trình mà phòng ban mình nhận được rồi mời bạn tính lại thử xem!

Yêu cầu:

Cho x, a, y, b, n. Hãy tính tổng giá trị máy tính mà phòng Thắng nhận được.

Input:

Gồm hai bộ dữ liệu, mỗi bộ trên một dòng, mỗi dòng chứa 5 số nguyên dương x, a, y, b, n (các số có giá trị không vượt quá 1000).

Output:

Gồm hai dòng, mỗi dòng là đáp án tương ứng với bộ dữ liệu vào.

Ví dụ:

Input	Output
3 300 2 500 2	900
4 300 3 500 2	1300

Lời giải

Trước hết ta sẽ chặt nhị phân kết quả bài toán. Với mỗi giá trị chặt nhị phân, ta cần kiểm tra xem có tồn tại phương án thỏa mãn hay không.

Thuật toán sơ khai

Đặt giá trị cần kiểm tra là v.

Xét các phòng ban theo thứ tự tăng dần về mức độ quan trọng, đánh số từ 1.

Sử dụng một mảng đa chiều để đánh dấu các trạng thái có thể đạt tới. Các giá trị cần quản lí là: chỉ số của phòng ban, đã dùng số máy tính để bàn x, đã dùng số máy tính xách tay y, tổng giá trị máy tính của phòng ban trước đó.

Bắt đầu từ trạng thái (0, 0, 0, 0), ta sử dụng thuật toán loang (BFS). Cuối cùng nếu trạng thái (n, 0, 0, ...) có thể đến được, thì ta sẽ có cách phân hoạch các máy tính vào các phòng ban ứng với giá trị cận dưới v.

Không cần tính toán cụ thể cũng có thể thấy thuật toán này không thể đáp ứng về mặt thời gian (và bộ nhớ) với giới hạn của đề bài.

Nâng cấp bằng nhận xét

Nhận xét rằng ta *không cần quan tâm tới thứ tự về mức độ quan trọng của các phòng ban*. Với một cách phân hoạch các máy tính sao cho mỗi phòng nhận được tổng giá trị không nhỏ hơn v, ta luôn có thể sắp xếp các bộ theo giá trị không giảm ứng với các phòng ban.

Ta có trạng thái QHĐ là F(i, x, y, value) = true nếu có thể phân bổ máy tính cho i phòng ban, đã dùng x máy tính để bàn và y máy tính xách tay, đã gom được tổng giá trị v cho phòng thứ i+1. Cách làm này số trạng thái vẫn như trước nhưng ta đã có thể chuyển trạng thái trong O(1). Cụ thể từ F(i, x, y, value) ta chuyển đến F(i, x+1, y, value+a) hoặc F(i, x, y+1, value+b), chú ý là chỉ có thể dùng thêm máy xách tay nếu x<X và dùng thêm máy để bàn nếu y<Y, đồng thời nếu giá trị value đủ lớn hơn hoặc bằng v thì ta chuyển sang trạng thái F(i+1, x, y, 0) luôn.

Đổi biến

Ở bài này, ta có thể dễ dàng đổi biến value ra làm hàm mục tiêu. Nhưng không chỉ có vậy, ta có thể đẩy cả i ra ngoài! Cụ thể, F(x, y) = một cặp số (i, value) lần lượt là số phòng phân bố được và số tiền gom được. Hàm mục tiêu của F(x, y) là một cặp số hoàn toàn có thể so sánh được, trong đó giá trị đầu (i) được ưu tiên so sánh trước.

Cách cập nhật các F(x, y) giống như phần trước, độ phức tạp vẫn là O(1) cho bước chuyển trạng thái, trong khi số trạng thái lúc này là đủ nhỏ đối với giới hạn của đề bài.

```
#include <bits/stdc++.h>

using namespace std;

const int N = 1010;

int x, y, a, b, n;

pair<int, int> F[N][N];

pair<int, int> newState(pair<int, int> s, int a, int v) {
s.second += a;
if (s.second >= v) {
++s.first;
s.second = 0;
}
return s;
}

bool dp(int value) {
for (int i = 0; i <= x; ++i) for (int j = 0; j <= y; ++j)
F[i][j] = make_pair(0, 0);
for (int i = 0; i <= x; ++i) for (int j = 0; j <= y; ++j) {
```

```
if (F[i][j].first == n) return 1;
if (i < x)
F[i + 1][j] = max(F[i + 1][j], newState(F[i][j], a, value));
if (j < y)
F[i][j + 1] = max(F[i][j + 1], newState(F[i][j], b, value));
}
return 0;
}

int solve() {
int l = 0, r = (a * x + b * y) / n;
int ans = 0;
while (l <= r) {
int mid = l + r >> 1;
if (dp(mid)) {
ans = mid;
l = mid + 1;
} else {
r = mid – 1;
}
}
return ans;
}

int main() {
cin >> x >> a >> y >> b >> n;
cout << solve() << endl;
cin >> x >> a >> y >> b >> n;
cout << solve() << endl;
return 0;
}
```

Bài luyện tập: http://vn.spoj.com/problems/BINPACK/

2. Chia để trị

Đây là kỹ thuật khá hiếm gặp, tuy nhiên lại cực kỳ mạnh.

Bài tập ví dụ

Hai nhà máy (CEOI 2004)

Có n cây cổ thụ được trồng trên một con đường từ đỉnh đổi đến chân đồi. Chính phủ địa phương quyết định cắt bỏ chúng. Để tránh hoang phí, mỗi cái cây cần được chuyển đến một nhà máy cưa.

Cây chỉ có thể được vận chuyển theo một chiều duy nhất: hướng về chân đồi. Có một nhà máy cưa ở cuối con đường. Hai nhà máy cưa có thể được xây dựng dọc theo con đường. Hãy xác định vị trí tối ưu để xây dựng chúng, để cực tiểu hóa chi phí vận chuyển. Chi phí vận chuyển 1kg gỗ đi 1 mét là 1 cent.

Yêu cầu

Viết chương trình:

- **đọc dữ liệu từ đầu vào chuẩn số lượng cây, khối lượng và vị trí của chúng,**
- tính toán chi phí vận chuyển tối ưu nhất,
- xuất kết quả ra đầu ra chuẩn.

Input

Dòng đầu tiên chứa số n – số lượng cây (2 <= n <= 20000). Các cây được đánh số 1, 2, …, n, theo chiều từ đỉnh đồi đến chân đồi.

n dòng tiếp theo mỗi dòng chứa hai số nguyên dương cách nhau bởi dấu cách. Dòng thứ i + 1 chứa w(i) – khối lượng tính theo kg của cái cây thử i và d(i) – khoảng cách tính theo mét giữa cây thứ i và cây i + 1, 1 <= w(i) <= 10000, 0 <= d(i) <= 10000. Số cuối cùng, d(n) là khoảng cách từ cây thứ n đến chân đồi.

Dữ liệu vào đảm bảo kết quả của bài toán không vượt quá 2 000 000 000 cent.

Output

Một dòng duy nhất chứa một số là kết quả bài toán: chi phí vận chuyển nhỏ nhất.

Ví dụ

Input	Output
9 1 2 2 1 3 3 1 1 3 2 1 6 2 1 1 2 1 1	26

Hình vẽ trên minh họa cho test ví dụ. Các hình tròn được tô đen là các vị trí có nhà máy. Kết quả sẽ là:
1 * (2+1) + 2 * 1 + 1 * (1 + 2) + 3 * 2 + 2 * (1 + 2 + 1) + 1 * (2 + 1) + 1 * 1 = 26.

Lời giải

Trước hết ta sẽ giải quyết vấn đề tính chi phí vận chuyển nếu biết vị trí của hai nhà máy đặt thêm.

Nếu ta có thể tính được chi phí này trong O(1), bài toán sẽ có thể giải được trong O(N^2) – ta có thể for hết các cặp vị trí có thể đặt nhà máy.

Gọi:

- **sumW(i) = tổng của các w(j) với i <= j.**
- sumD(i) = tổng của các d(j) với i <= j.
- sumWS(i) = tổng của các w(j) * sumD(j) với i <= j.

Khi đó cost(L, R) = chi phí vận chuyển các cây có chỉ số trong đoạn [L..R] đến nhà máy đặt ở R là: sumWS(L) – sumWS(R) – sumD(R) * (sumW(L) – sumW(R)).

Như vậy ta có thể xây dựng hàm eval(i, j) = chi phí nếu đặt thêm hai nhà máy ở i và j = cost(1, i) + cost(i + 1, j) + cost(j + 1, n + 1).

Tuy nhiên lời giải O(N^2) là chưa đủ tốt để có thể giải quyết trọn vẹn bài toán này.

Gọi best(i) = vị trí j > i tốt nhất nếu ta đã đặt một nhà máy ở i.

Như vậy kết quả của bài toán sẽ là min(eval(i, best(i)) với 1 <= i < n.

Nhận xét:

- **best(i) <= best(i + 1).**
- Ta có thể tính các best(i) theo thứ tự bất kỳ.

Như vậy ta có thuật toán sử dụng tư tưởng chia để trị như sau:

Hàm solve(L, R, from, to) sẽ đi tính các best(L..R), biết rằng chúng nằm trong đoạn [from..to].

```
void solve(int L, int R, int from, int to) {
if (L > R) return;
int mid = L + R >> 1;
best[mid] = from;
for (int i = from + 1; i <= to; ++i)
if (eval(mid + 1, best[mid]) > eval(mid + 1, i))
best[mid] = i;
solve(L, mid – 1, from, best[mid]);
solve(mid + 1, R, best[mid], to);
}
```

Đánh giá độ phức tạp thuật toán: vì mỗi lần gọi để quy khoảng L..R được chia đôi, nên sẽ có O(logN) tầng, mỗi tầng vòng for chỉ chạy qua O(N) phần tử, vì vậy độ phức tạp của thuật toán là O(NlogN).

Bài tập ví dụ

SEQPART

Submit: hackerrank.com

Cho dãy L số C[1..L], cần chia dãy này thành G đoạn liên tiếp. Với phần tử thứ i, ta định nghĩa chi phí của nó là tích của C[i] và số lượng số nằm cùng đoạn liên tiếp với nó. Chi phí của dãy số ứng với một cách phân hoạch là tổng các chi phí của các phần tử.

Hãy xác định cách phân hoạch dãy số để chi phí là nhỏ nhất.

Input:

- **Dòng đầu tiên chứa 2 số L và G.**
- L dòng tiếp theo, chứa giá trị của dãy C.

Output:

- **Một dòng duy nhất chứa chi phí nhỏ nhất.**

Giới hạn:

- **1 <= L <= 8000**
- 1 <= G <= 800
- 1 <= C[i] <= 1 000 000 000

Ví dụ:

Input	Output
6 3 11 11 11 24 26 100	299

Giải thích: cách tối ưu là C[] = (11, 11, 11), (24, 26), (100). Chi phí là 11 * 3 + 11 * 3 + 11 * 3 + 24 * 2 + 26 * 2 + 100 * 1 = 299.

Lời giải

Đây là dạng bài toán phân hoạch dãy số có thể dễ dàng giải bài QHĐ. Gọi F(g, i) là chi phí nhỏ nhất nếu ta phân hoạch i phần tử đầu tiên thành g nhóm, khi đó kết quả bài toán sẽ là F(G, L).

Để tìm công thức truy hồi cho hàm F(g, i), ta sẽ quan tâm đến nhóm cuối cùng. Coi phần tử 0 là phần tử cầm canh ở trước phần tử thứ nhất, thì người cuối cùng không thuộc nhóm cuối có chỉ số trong đoạn [0..i]. Giả sử đó là người với chỉ số k, thì chi phí của cách phân hoạch sẽ là F(g-1, k) + Cost(k+1, i), với Cost(i, j) là chi phí nếu phân j-i+1 người có chỉ số [i..j] vào một nhóm. Như vậy:

F(g, i) = min[F(g-1, k) + Cost(k+1, I)] với 0 <= k <= i.

Chú ý là công thức này chỉ được áp dụng với g>1, nếu g=1, F(1, i) = Cost(1, i), đây là trường hợp cơ sở.

Việc cài đặt chỉ đơn giản là dựng mảng 2 chiều F[][], code như sau:

```
#include <iostream>
```

```
using namespace std;

const int MAXL = 8008;
const int MAXG = 808;
const long long INF = (long long)1e18;

long long C[MAXL];
long long sum[MAXL];
long long F[MAXG][MAXL];

long long cost(int i, int j) {
return (sum[j] – sum[i – 1]) * (j – i + 1);
}

int main() {
int G, L;
cin >> L >> G;
for (int i = 1; i <= L; ++i) {
cin >> C[i];
sum[i] = sum[i – 1] + C[i];
}

for (int g = 1; g <= G; ++g) {
for (int i = 0; i <= L; ++i) {
if (g == 1) {
F[g][i] = cost(1, i);
} else {
F[g][i] = INF;
for (int k = 0; k <= i; ++k) {
long long new_cost = F[g – 1][k] + cost(k + 1, i);
if (F[g][i] > new_cost) F[g][i] = new_cost;
}
}
}
}
cout << F[G][L] << endl;
return 0;
}
```

Chú ý là ta sử dụng mảng sum[] tiền xử lí O(L) để có thể truy vấn tổng một đoạn (dùng ở hàm cost()) trong O(1). Như vậy độ phức tạp của thuật toán này là O(G*L*L).

Thuật toán tối ưu hơn

Gọi P(g, i) là k nhỏ nhất để cực tiểu hóa F(g, i), nói cách khác, P(g, i) là k nhỏ nhất mà F(g, i) = F(g-1, k) + Cost(k+1, i).

Tính chất quan trọng để có thể tối ưu thuật toán trên là dựa vào tính đơn điệu của P(g, i), cụ thể:

P(g, 0) <= P(g, 1) <= P(g, 2) <= … <= P(g, L-1) <= P(g, L)

Ta sẽ không chứng minh điều này ở đây, độc giả có thể tự thuyết phục rằng điều này là đúng.

Chia để trị

Để ý rằng để tính F(g, i), ta chỉ cần quan tâm tới hàng trước <F(g-1)> của ma trận:

F(g-1, 0), F(g-1, 1), … , F(g-1, L). Như vậy, ta có thể tính hàng F(g) theo thứ tự bất kỳ.

Ý tưởng là với hàng g, trước hết ta tính F(g, mid) và P(g, mid) với mid=L/2, sau đó sử dụng tính chất nêu trên P(g, i) <= P(g, mid) với i < mid và P(g, i) >= P(g, mid) với i > mid để đi gọi đệ quy đi tính hai nửa còn lại.

```
#include <iostream>

const int MAXL = 8008;
const int MAXG = 808;
const long long INF = (long long)1e18;

using namespace std;

long long F[MAXG][MAXL], sum[MAXL], C[MAXL];
int P[MAXG][MAXL];

long long cost(int i, int j) {
if (i > j) return 0;
return (sum[j] – sum[i – 1]) * (j – i + 1);
}

void solve(int g, int L, int R, int optL, int optR) {
if (L > R) return;
int mid = (L + R) / 2;
F[g][mid] = INF;
for (int i = optL; i <= optR; ++i) {
long long new_cost = F[g – 1][i] + cost(i + 1, mid);
if (F[g][mid] > new_cost) {
F[g][mid] = new_cost;
P[g][mid] = i;
}
}
solve(g, L, mid – 1, optL, P[g][mid]);
solve(g, mid + 1, R, P[g][mid], optR);
}
```

```
int main() {
int G, L;
cin >> L >> G;
for (int i = 1; i <= L; ++i) {
cin >> C[i];
sum[i] = sum[i – 1] + C[i];
}
for (int i = 1; i <= L; ++i) F[1][i] = cost(1, i);
for (int g = 2; g <= G; ++g) solve(g, 1, L, 1, L);
cout << F[G][L] << endl;
return 0;
}
```

Chú ý rằng ta không thể đảm bảo rằng P[g][mid] chia đôi đoạn [optL..optR], thực tế một vài hàm solve() sẽ chạy chậm hơn nhiều hàm solve() khác.

Tuy nhiên ta có thể chứng minh được, xét về tổng thế thuật toán này chạy đủ nhanh. Mỗi lần ta chia đôi đoạn [L..R], nên ta sẽ đảm bảo có tối đa O(log(L)) tầng đệ quy, như vậy với mỗi hàng g, ta chỉ mất O(L logL) để tính. Toàn bộ thuật toán có độ phức tạp là O(GLlogL).

Bài luyện tập: https://www.hackerrank.com/contests/world-codesprint-5/challenges/mining

CHỦ ĐỀ: PHƯƠNG PHÁP , QUY HOẠCH ĐỘNG

Khuyên dùng

About Aida Nana
Nghề chính là chém gió, quăng bom và ném lựu đạn.
Nghề phụ là cắt cỏ, chém chuối, cưa cây……

Speak Your Mind

Name *

Email *

Website

PHẢN HỒI

Đôi điều về YeuLapTrinh.pw

Website được viết bởi nhiều thành viên trên mọi miền tổ quốc với mục đích chia sẻ, trao đổi, giúp đỡ lẫn nhau trong học lập trình. Hi vọng được sự ủng hộ của các bạn đọc thân mến <3

Các chủ đề:

- Thuật toán
- Cấu trúc dữ liệu
- Ngôn ngữ lập trình
- Lập trình Web
- Thủ thuật máy tính
- Kiếm tiền Online

Sử dụng Website

Viết bài:

Để viết bài bạn phải đăng ký thành viên. Bài viết bạn viết sẽ được kiểm duyệt và đăng lên trang Web

Mọi vẫn đề thắc mắc, đóng góp xin liên hệ:

yeulaptrinh.pw@gmail.com

RETURN TO TOP OF PAGE