

Thuật toán ứng dụng

Bài thực hành số 1.2: Cấu trúc dữ liệu

TS. Bùi Quốc Trung, TA. Đặng Xuân Vương



Trường Đại học Bách khoa Hà Nội
Viện Công nghệ thông tin và Truyền thông

Ngày 16 tháng 3 năm 2021

Mục lục

1 HIST

2 REROAD

3 SIGNAL

Mục lục

1 HIST

2 REROAD

3 SIGNAL

- Có N cột với độ cao l_1, l_2, \dots xếp liên tiếp.
- Tìm diện tích hình chữ nhật lớn nhất nằm gọn trong N cột.

- Nhận xét: Hình chữ nhật có 2 cột biên i, j sao cho $i < j$ có diện tích là $(j - i + 1) \times \min(l_i, \dots, l_j)$.
- Thuật toán: Thử hết các cặp (i, j) , duyệt từ i đến j để tìm cột thấp nhất (tương tự bài SUBSEQMAX).
- Độ phức tạp: $O(n^3)$.

- Nhận xét: $\min(l_i, \dots, l_j) = \min(\min(l_i, \dots, l_{j-1}), l_j)$.
- Cải tiến: $O(n^2)$.

Hướng tiếp cận khác

- Chọn cột i làm cột thấp nhất trong hình chữ nhật.
- Tìm $left_i$ là cột gần nhất bên trái i thấp hơn cột i .
- Tìm $right_i$ là cột gần nhất bên phải i thấp hơn cột i .
- Số cột trong hình chữ nhật là $right_i - left_i - 1$.
- Độ phức tạp: $O(n^2)$.

Tăng tốc bằng ngăn xếp

- Với left, duyệt các cột từ trái sang phải, khi duyệt đến cột i :
 - Loại cột ở đầu stack trong khi nó còn cao hơn cột i .
 - $left_i$ là cột ở đầu stack hiện tại.
 - Độ phức tạp $O(N)$.
- Tương tự với right, chỉ khác là duyệt từ phải sang trái.
- Độ phức tạp tổng: $O(N)$.


```
template <class RandomIt>
vector<int> calc_extend(RandomIt first , RandomIt last)
{
    vector<int> result;
    stack <RandomIt> s ;
    for (RandomIt it = first; it != last; it++) {
        while (!s.empty() && *s.top() >= *it) s.pop();
        result.push_back( it - (s.empty()
            ? (first-1) : s.top ()));
        s.push(it);
    }
    return result;
}
```

```
int main() {
    int n;
    while ((cin >> n) && n) {
        vector<long long> l(n);
        for (int i = 0; i < n; i++) cin >> l[i];
        vector<int> L = calc_extend(l.begin(), l.end());
        vector<int> R = calc_extend(l.rbegin(), l.rend());
        long long result = 0;
        for (int i = 0; i < n; ++ i ) {
            result = max(result, (L[i]+R[n-i-1]-1)*l[i]);
        }
        cout << result << endl ;
    }
}
```

Mục lục

1 HIST

2 REROAD

3 SIGNAL

02. REROAD

- Cho N đoạn đường, đoạn đường i dùng phủ nhựa đường loại t_i .
- Một phần đường là:
 - Dây liên tục các đoạn đường dùng chung loại nhựa đường t_k .
 - Đoạn đường phía trước đoạn đường đầu dây (nếu có) không dùng nhựa đường loại t_k .
 - Tương tự với đoạn đường phía sau đoạn đường cuối dây.
- Độ gập ghềnh là số phần đường.
- Yêu cầu: Tính lại độ gập ghềnh sau khi một đoạn đường bị sửa.

Ví dụ



- Nhận xét: $t[i] \neq t[i - 1]$ thì i chính là điểm bắt đầu phần đường.
- Số phần đường bằng số điểm bắt đầu phần đường.

- Khi thay đổi $t[i]$, sự thay đổi chỉ xảy ra ở i và $i + 1$.

```
int isStart(int u) {
    return u == 1 ? 1 : t[u] != t[u-1];
}

int main() {
    cin >> N;
    for (int i = 1; i <= N; i++) cin >> t[i];
    int res = 0;
    for (int i = 1; i <= N; i++) res += isStart(i);
    cin >> Q;
    while (Q--) {
        cin >> p >> c;
        res -= isStart(p);
        if (p < N) res -= isStart(p+1);
        t[p] = c;
        res += isStart(p);
        if (p < N) res += isStart(p+1);
        cout << res << endl;
    }
}
```


Mục lục

1 HIST

2 REROAD

3 SIGNAL

02. SIGNAL

- Cho tín hiệu độ dài n có độ lớn a_1, a_2, \dots, a_n và giá trị phân tách b .
- Tín hiệu phân tách được tại i nếu:
 - $\max\{a_1, \dots, a_{i-1}\} - a_i \geq b$
 - $\max\{a_{i+1}, \dots, a_n\} - a_i \geq b$
- Tìm vị trí i để tín hiệu phân tách được và $\max\{a_1, \dots, a_{i-1}\} - a_i + \max\{a_{i+1}, \dots, a_n\} - a_i$ lớn nhất.

- Xây dựng mảng $maxPrefix[i] = \max\{a_1, \dots, a_i\}$
- Xây dựng mảng $maxSuffix[i] = \max\{a_i, \dots, a_n\}$
- Duyệt từng vị trí i , kiểm tra điều kiện phân tách và cập nhật giá trị.
- Độ phức tạp: $O(n)$.

```
long long solve() {
    maxPrefix[0] = -INF; maxSuffix[n+1] = -INF;
    for (int i = 1; i <= n; i++)
        maxPrefix[i] = max(maxPrefix[i-1], a[i]);
    for (int i = n; i >= 1; i--)
        maxSuffix[i] = max(maxSuffix[i+1], a[i]);
    long long res = -1;
    for (int i = 2; i < n; i++)
        res = check(i) ? max(res, val(i)) : res;
    return res;
}
```

Thuật toán ứng dụng

Bài thực hành số 1.2: Cấu trúc dữ liệu

TS. Bùi Quốc Trung, TA. Đặng Xuân Vương



Trường Đại học Bách khoa Hà Nội
Viện Công nghệ thông tin và Truyền thông

Ngày 16 tháng 3 năm 2021