Note book

Cây phân đoạn

Dưới đây là một ví dụ cụ thể về cách sử dụng cây phân đoạn để giải quyết bài toán bạn đã đề cập:

```
Để tìm giá trị lớn nhất từ x đến y trong một vector sử dụng cây phân đoạn, chúng ta
```

```
#include <iostream>
#include <vector>
using namespace std;

struct SegmentTreeNode {
    int start, end, maxVal;
    SegmentTreeNode *left, *right;

    SegmentTreeNode(int start, int end) {
        this->start = start;
        this->end = end;
        this->maxVal = 0;
        this->left = this->right = nullptr;
    }
};

SegmentTreeNode *build_segment_tree(vector<int> &nums, int start, int end) {
    if (start > end) {
        return nullptr;
    }

    auto root = new SegmentTreeNode(start, end);

    if (start == end) {
        root->maxVal = nums[start];
        return root;
    }
```

```
    int mid = (start + end) / 2;
    root->left = build_segment_tree(nums, start, mid);
    root->right = build_segment_tree(nums, mid + 1, end);
    root->maxVal = max(root->left->maxVal, root->right->maxVal);

    return root;
}

int query_segment_tree(SegmentTreeNode *root, int start, int end) {
    if (root->start == start && root->end == end) {
        return root->maxVal;
    }

    int mid = (root->start + root->end) / 2;

    if (end <= mid) {
        return query_segment_tree(root->left, start, end);
    } else if (start > mid) {
        return query_segment_tree(root->right, start, end);
    } else {
        int left_max = query_segment_tree(root->left, start, mid);
        int right_max = query_segment_tree(root->right, mid + 1, end);
        return max(left_max, right_max);
    }
}

int main() {
    vector<int> nums = {10, 5, 8, 12, 3, 7, 15, 9};
    int x, y;
    cout << "Nhap gia tri x: ";
    cin >> x;
    cout << "Nhap gia tri y: ";
    cin >> y;

    auto root = build_segment_tree(nums, 0, nums.size() - 1);

    int maxVal = query_segment_tree(root, x, y);

    cout << "Gia tri lon nhat tu " << x << " den " << y << " la: " << maxVal << enc

    return 0;
```

```
}
```

Trong ví dụ trên, chúng ta sử dụng cây phân đoạn để lưu trữ giá trị lớn nhất của mỗi đoạn. Ban đầu, tất cả các phần tử trong dãy có giá trị 0. Sau đó, chúng ta thực hiện từng truy vấn theo yêu cầu. Nếu truy vấn là loại 1, chúng ta gán giá trị v cho phần tử ở vị trí i. Nếu truy vấn là loại 2, chúng ta tìm giá trị lớn nhất trong đoạn [i, j] bằng cách truy vấn cây phân đoạn.

Tổng tiền tố trên ma trận

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

using namespace std;

long long query(int x1, int y1, int x2, int y2,
                vector < vector < long long > > &sum)
{
    return sum[x2][y2] - sum[x1 - 1][y2] - sum[x2][y1 - 1] + sum[x1 - 1][y1 - 1];
}

main()
{
    int m, n, q;
    cin >> m >> n >> q;

    vector < vector < long long > > sum(m + 1, vector < long long >(n + 1, 0));
    for (int i = 1; i <= m; ++i)
        for (int j = 1; j <= n; ++j)
        {
            int x;
            cin >> x;

            sum[i][j] = sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j - 1] + x;
        }

    while (q--)
```

```
    {
        int x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;

        cout << query(x1, y1, x2, y2, sum) << endl;
    }
}
```

đếm cách trả tiền

```
#include <bits/stdc++.h>
using namespace std;
#define endl "\n"
const int ll = 1e6, oo = 1e9, mod = 1e9 + 7;
int dp[101][ll + 1],t[ll], n, m;
void enter()
{
    cin >> n >> m;
    for(int  i = 1; i <=n; i++)
    {
        cin >> t[i];
    }
}

void build()
{
    dp[0][0] = 1;
    for (int i = 1; i <= n; i++){
        for (int j = 0; j <= m; j++)
        {
            dp[i][j] = dp[i - 1][j];
            if (j >= t[i])
                dp[i][j] = (dp[i][j] + dp[i][j - t[i]]) % mod;
        }
    }

    cout << dp[n][m];
}
main()
{
```

```
    ios_base::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    enter();
    build();

}
```