

Lazy Propagation in Segment Tree

CS523.011 - Group 6

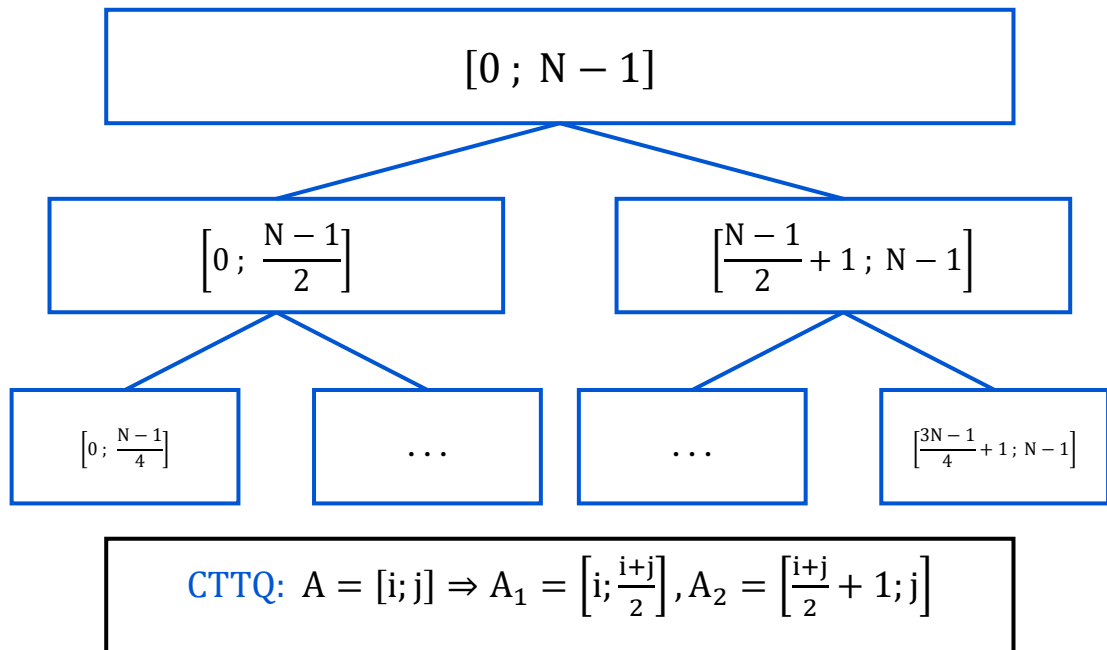
Nguyễn Hoàng Xuân Huy
21520924

Trần Ngọc Thiện
21521465

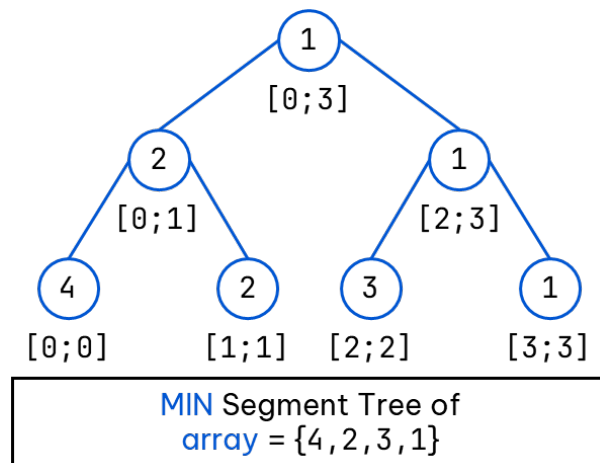
1. Introduction

a) What is Segment Tree?

- » Segment Tree (Cây Phân Đoạn) là 1 CTDL dạng cây, thường được sử dụng trong các bài toán cần xử lý trên array.
- » Segment Tree được xây dựng dựa trên phương pháp “Chia để Trị” với mỗi nút trên cây quản lý một đoạn trên array:



- » Các nút này chứa dữ liệu về kết quả của một phép toán hay phép truy vấn nào đó trên đoạn mà nó quản lý, ví dụ như:
 - > Cộng, Trừ, Nhân, Chia, ...
 - > Tìm Max, tìm Min, GCD, LCM, ...
 - > AND, OR, XOR, ...



b) Segment Tree Implementation (C++)

Segment Tree thường được cài đặt bằng array với 2 cách cài đặt: **độ quy** và **không độ quy**. Sau đây là ví dụ về 2 cách cài đặt trên cho Sum Segment Tree:

» **Khai báo:**

Cách 1	Cách 2
<pre> const int N = 100000; int arr[N]; int tree[N * 4]; // Build void build(int node, int l, int r) { if (l == r) tree[node] = arr[l]; else { int mid = (l + r) / 2; build(2 * node, l, mid); build(2 * node + 1, mid + 1, r); tree[node] = tree[2 * node] + tree[2 * node + 1]; } } // Query int query(int node, int l, int r, int ql, int qr) { if (ql >= l & amp; qr <= r) return tree[node]; int mid = (l + r) / 2; int res = 0; if (ql <= mid) res += query(2 * node, l, mid, ql, qr); if (qr > mid) res += query(2 * node + 1, mid + 1, r, ql, qr); return res; } </pre>	<pre> const int N = 100000; int arr[N]; int tree[N * 4]; // Build void build(int node, int l, int r) { if (l == r) tree[node] = arr[l]; else { int mid = (l + r) / 2; build(2 * node, l, mid); build(2 * node + 1, mid + 1, r); tree[node] = tree[2 * node] + tree[2 * node + 1]; } } // Query int query(int node, int l, int r, int ql, int qr) { if (ql >= l & amp; qr <= r) return tree[node]; int mid = (l + r) / 2; int res = 0; if (ql <= mid) res += query(2 * node, l, mid, ql, qr); if (qr > mid) res += query(2 * node + 1, mid + 1, r, ql, qr); return res; } </pre>

» **Nhập dữ liệu cho arr (optional):**

<pre> // Input for (int i = 0; i < N; i++) { int x; cin >> x; arr[i] = x; } </pre>

» **Build Segment Tree**

<p>Cách 1</p> <pre> // Build void build(int node, int l, int r) { if (l == r) tree[node] = arr[l]; else { int mid = (l + r) / 2; build(2 * node, l, mid); build(2 * node + 1, mid + 1, r); tree[node] = tree[2 * node] + tree[2 * node + 1]; } } </pre>	
--	--

Giải thích:

- > $ST[0]$ nằm ở trên đỉnh là tổng của cả đoạn $[0; n-1]$
- > Node con của node thứ i là $(2i+1)$ và $(2i+2)$
VD: node con của node root là $ST[1]$ và $ST[2]$, node con của $ST[1]$ là $ST[3]$ và $ST[4]$, ...
- > [Example1.pptx](#)

<p>Cách 2</p>	<p>Để tính tổng của đoạn $[L; R]$ ta sẽ đi tìm các node mà đoạn $[L; R]$ nằm trong nó. Đầu tiên ta tìm node u mà $[L; R]$ nằm trong nó. Sau đó ta đi tìm các node con của u mà $[L; R]$ nằm trong nó. Cuối cùng ta tính tổng của các node đó.</p>
----------------------	---

Giải thích: Tương tự như trên:

- > $ST[0]$ nằm ở trên đỉnh là tổng của cả đoạn
- > Node con của node thứ i là $(2i+1)$ và $(2i+2)$
- > [Example2.pptx](#)

» Query [L;R]

<p>Cách 1</p>	<p>Để tính tổng của đoạn $[L; R]$ ta sẽ đi tìm các node mà đoạn $[L; R]$ nằm trong nó. Đầu tiên ta tìm node u mà $[L; R]$ nằm trong nó. Sau đó ta đi tìm các node con của u mà $[L; R]$ nằm trong nó. Cuối cùng ta tính tổng của các node đó.</p>
----------------------	---

c) Segment Tree Implementation (Python)

