

# Drzewo Przedziałowe

Struktura danych, która pozwala na aktualizowanie wartości w tablicy w danym punkcie oraz sprawdzanie wartości tablicy w danym przedziale w czasie  $O(\log n)$

Standardowy Problem:

Dana jest tablica  $T[0, n - 1]$ . Chcemy  $q$  razy wykonywać dwa typy operacji:

1. Ustaw  $T[i] = x$ .
2. Zwróć sumę na przedziale  $T[a, b]$ . Formalnie:  $\sum_{i=a}^b T[i]$

Limity:

$$1 \leq n, q \leq 2 \cdot 10^5$$

$$1 \leq T[i], x \leq 10^3$$

Rozwiązanie w czasie  $O(q \log n)$ :

Budujemy drzewo przedziałowe, które jest drzewem binarnym o takiej własności, że każdy wierzchołek obejmuje pewien spójny przedział tablicy  $T$ :

- Korzeń obejmuje przedział  $T[0, n - 1]$ .
- Potem lewy syn korzenia obejmuje  $T[0, \frac{n}{2} - 1]$ , a prawy  $T[\frac{n}{2}, n - 1]$ .
- Przedziały dzielimy tak długo, jak ich rozmiar jest większy od 1.
- Na koniec liście obejmują przedziały długości 1, czyli poszczególne komórki  $T$ .

Dzięki takiej budowie wysokość drzewa wynosi  $\log n$ . Bo za każdym razem dzielimy przedział na 2 równe połowy.

Teraz w każdym wierzchołku przechowujemy sumę wartości z  $T$  na przedziale, który on obejmuje.

## Jak przechowywać drzewo w pamięci?

Nasze drzewo będziemy przechowywać w jednej tablicy  $Tree[0, 2 \cdot base]$ , gdzie  $base$  to dowolna potęga 2 nie mniejsza od  $n$ .

Teraz:

- Korzeń drzewa ma indeks 1
- Jeśli mamy wierzchołek w drzewie o indeksie  $v$ , to:
  - lewy syn ma indeks  $2v$
  - prawy syn ma indeks  $2v + 1$
- Wszystkie wierzchołki o indeksach większych bądź równych  $base$  to liście

## Jak wykonać operację 1) ?

1. Ustaw  $x$  w liściu, który obejmuje  $i$ -tą komórkę  $T$

2. Odśwież sumę we wszystkich wierzchołkach na ścieżce z danego liścia do korzenia drzewa.

```
void update(int v, int val) {
    v += base;
    Tree[v] = val;
    while (v) {
        v /= 2;
        Tree[v] = Tree[2*v] + Tree[2*v+1];
    }
}
```

### Jak wykonać operację 2) ?

Zauważmy, że każdy przedział  $[a, b]$  może być rozłożony na sumę (zbiorów) nie więcej niż  $\log n$  wielu mniejszych parami rozłącznych przedziałów. Te przedziały będą obejmowane przez pewne wierzchołki w drzewie. Zatem wystarczy znaleźć te wierzchołki i dodać do siebie wartości które przechowują.

```
int query(int a, int b) {
    int res = 0;
    a += base - 1;
    b += base + 1;
    while (a / 2 != b / 2) {
        if (a % 2 == 0) res += Tree[a+1];
        if (b % 2 != 0) res += Tree[b-1];
        a /= 2;
        b /= 2;
    }
    return res;
}
```

Obie operacje wykonujemy w czasie  $O(\log n)$  zatem złożoność całego rozwiązania to  $O(q \log n)$ .