Drzewo Przedziałowe Punkt-Przedział

Autor: Franciszek Pietrusiak

Czym jest?

Drzewo przedziałowe to struktura danych, która pozwala na aktualizowanie wartości w danym punkcie oraz sprawdzanie wartości na całym przedziale w czasie $O(\log n)$ (lub sprawdzenie wartości w punkcie oraz aktualizacja wartości na całym przedziale)

Standardowe Zadanie

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*base]`, gdzie `base` to potęga 2 taka, że `base` $\geq n$.
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 aktualizować wartość w punkcie?

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];
        }
}
```

Przechodzimy wysokość drzewa dlatego złożoność czasowa funkcji `update` to $O(\log n)$.

Jak sprawdzić wartość na przedziale?

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;
}
```

Operacja `query` także działa w złożoności $O(\log n)$.

Co gdy chcemy sprawdzać wartość w punkcie i aktualizować przedział?

Aby to zrobić wystarczy nieznacznie zmodyfikować funkcje `update` i `query`.

Modyfikacja funkcji `query`

```
int query(int v) {
        int res = 0;
        v += base;
        while (v) {
                res += Tree[v];
                v /= 2;
        }
        return res;
}
```

Modyfikacja funkcji `update`

```
void update(int a, int b, int x) {
        a += base - 1;
```

```
        b += base + 1;
        while (a / 2 != b / 2) {
                if (a % 2 == 0) Tree[a+1] = x;
                if (b % 2 != 0) Tree[b-1] = x;
                a /= 2;
                b /= 2;
        }
}
```

Jakie operacje matematyczne można wykonywać na drzewie przedziałowym?

Aby operacja $\circ$ miała zastosowanie w drzewie przedziałowym musi ona być **łączna**. To znaczy, że zachodzi $(a \circ b) \circ c = a \circ (b \circ c)$. Dzięki łączności jesteśmy w stanie scalać przedziały, co jest podstawą działania drzewa przedziałowego.

Lista operacji jakie można spotkać w zadaniach na drzewa przedziałowe:

- dodawanie
- mnożenie (także modulo)
- min / max
- gcd / lcm
- logiczny and / or
- składanie funkcji
- mnożenie macierzy

Zadanie 1

Skrót Treści

Mamy tablicę liczb całkowitych T $(-10^9 \leq T[i] \leq 10^9)$ długości n $(1 \leq n \leq 2 * 10^5)$. Dostajemy q zapytań $(1 \leq q \leq 2 * 10^5)$ dwóch typów:

1. Zapytanie postaci $[1, k, x]$ $(1 \leq k \leq n)$. Należy zmienić wartość w k-tej komórce tablicy.
2. Zapytanie postaci $[2, a, b]$ $(1 \leq a \leq b \leq n)$. Należy zwrócić jedną liczbę całkowitą oznaczającą sumę liczb w maksymalnym spójnym podciągu T, który zawarty jest w przedziale $[a, b]$. Ten pociąg może być pusty.

Żeby lepiej zrozumieć zapytanie 2). Wyobraźmy sobie, że mamy zapytanie o fragment tablicy T postaci: $(-10, 3, 7, 4, -6, 3)$. Odpowiedzią będzie 13 i jest to suma elementów z podciągu $(3, 7, 4)$.

Rozwiązanie

W każdym węźle drzewa przedziałowego będziemy trzymali 4 informacje odnośnie przedziału, który ten węzeł obejmuje:

- `suma` = suma liczb na przedziale
- `pref` = maksymalna suma liczb na prefiksie przedziału
- `suff` = maksymalna suma liczb na sufiksie przedziału
- `ans` = suma liczb w maksymalnym spójnym podciągu zawartym w przedziale

Zauważmy, że wyliczając te informacje od liści do korzenia jesteśmy w stanie łatwo aktualizować wynik dla kolejnych wierzchołków. Dla liści sytuacja jest trywialna, bo obejmują one przedział jednoelementowy. Załóżmy zatem, że chcemy zaktualizować ojca mając wyliczone informacje dla obu jego synów. Są tylko trzy możliwości:

1. maksymalny spójny podciąg zawiera się wyłącznie w lewym synu.
2. maksymalny spójny podciąg zawiera się wyłącznie w prawym synu.
3. maksymalny spójny podciąg zawiera się w obu synach. Wtedy suma jego elementów to `suff` lewego syna dodać `pref` prawego syna.

Wiedząc jak aktualizować wierzchołki reszta zadania sprowadza się do naklepania Drzewa Punkt-Przedział.

Zadania:

1. Tetris 2d
2. https://cses.fi/problemset/task/1651
3. https://cses.fi/problemset/task/2166/

Źródła użyte:

- https://kubin.w.staszic.waw.pl/sheets/fenwick-tree-best-tree.html
- https://www.quora.com/What-kind-of-operations-could-be-applied-over-a-segment-tree
- https://cp-algorithms.com/data_structures/segment_tree.html#finding-subsegments-with-the-maximal-sum