20190929

Binary Indexed Tree (BIT)

AtCoder C++ 競プロ データ構造 BIT 競プロライブラリ

Binary Indexed Tree

Binary Indexed Tree (以下、BIT) は以下を実現するデータ構造です。

- $sum(i)$sum(i):
  累積和 a_1a1 + a_2a2 + a_3a3 + ... + $a_i$aiを $O(\log N)$O(logN)で計算
- $add(i, x)$add(i,x):
  ii と xx が与えたれたとき $a_i$aiにxxを加算することを $O(\log N)$O(logN) で行う
- 要素数 NN に対してサイズ NNの配列を持つ

実装方法

累積和 $sum(i)$sum(i)

ii までの和を求めるには ii から $0$0まで、ii から最後の $1$1 ビットを減算しながら、ii の場所の値を加算していく。ii の最後の $1$1 ビットは `i & -i` と書く

値の更新 $add(i, x)$add(i,x)

i の値に xx を加えるためには、iから NN まで i に最後の 11 ビットを加算しながら、i の場所の値に xx を加算していく

ライブラリ

以下は C++ の BITのライブラリです。

```
template<typename T>
class BinaryIndexedTree {
    int N;
    vector<T> data;
public:
    BinaryIndexedTree(int N) : N(N) {
        data.resize(N + 1, 0);
    }

    T sum(int k) {
        T res = 0;
        for (; k > 0; k -= k & -k) {
            res += data[k];
        }
        return res;
    }

    void add(int k, T x) {
        for (; k <= N; k += k & -k) {
            data[k] += x;
        }
    }

    /**
     * v1 + v2 + … + vx ≧ W となる最小の x を求める
     */
    int lowerBound(int w) {
        if (w <= 0) return 0;
        int x = 0;
        int k = 1;
        while (k * 2 <= N) k *= 2;
        for (; k > 0; k /= 2) {
            if (x + k <= N && data[x + k] < w) {
                w -= data[x + k];
                x += k;
            }
        }
        return x + 1;
    }
};
```

ライブラリの動作確認に以下の問題が使えます。

AOJ: Range Sum Query

応用

二分探索の搭載

上記のライブラリの中にもありますが、二分探索を使って累積和が ww 以上となる最小の xx を求めるができます。 二分木の枝分かれに従い二分探索することで実現できます。

二次元配列のBIT

$H \times W$H×W の二次元配列に BIT を適用して累積和と値の更新を高速に行うことができます。要素数 WWの BIT を HH 個つくり BIT が BIT をつようにして管理します。

- $add(h, w, x)$add(h,w,x):
  V_hwVhw に値 xx を加える
- $sum(h, w)$sum(h,w):
  $(1, 1)$(1,1) から (h, w)(h,w) までの範囲の累積和を求める
- $sum(h1, w1, h2, w2)$sum(h1,w1,h2,w2):
  左上の座標 $(h1, w1)$(h1,w1) , 右下の座標 $(h2, w2)$(h2,w2) に含まれる値の累積和を求める

以下が 2 次元BIT のライブラリです。

```
template<typename T>
class BinaryIndexedTree2D {
    int H;
    int W;
    vector<vector<T> > data2d;
public:
    BinaryIndexedTree2D(int H, int W) : H(H), W(W) {
        data2d.resize(H + 1, vector<T>(W + 1, 0));
    }

    T sum(int h, int w) {
        T res = 0;
        for (int i = h; i > 0; i -= i & -i) {
            for (int j = w; j > 0; j -= j & -j) {
                res += data2d[i][j];
            }
        }
        return res;
    }

    /**
     * 左上の座標(h1, w1), 右下の座標(h2, w2) に含まれる値の累積和
     */
    T sum(int h1, int w1, int h2, int w2) {
        return sum(h2, w2) - sum(h1 - 1, w2) - sum(h2, w1 - 1) + sum(h1 -
    }
```

```
    void add(int h, int w, ll x) {
        for (int i = h; i <= H; i += i & -i) {
            for (int j = w; j <= W; j += j & -j) {
                data2d[i][j] += x;
            }
        }
    }
};
```

ライブラリの動作確認に以下の問題が使えます。

AOJ: Taiyaki-Master and Eater

BITで解くことができる問題

見つけ次第更新していきます。

- C - 積み木
- D. Distinct Characters Queries

参考

- Binary Indexed Tree のはなし
- AOJ 2842 たい焼きマスターと食べ盛り - けんちょんの競プロ精進記録
- Binary indexed tree