SCV/2021

Largest Rectangle in a Histogram

- Divide and Conquer, Segment Tree -

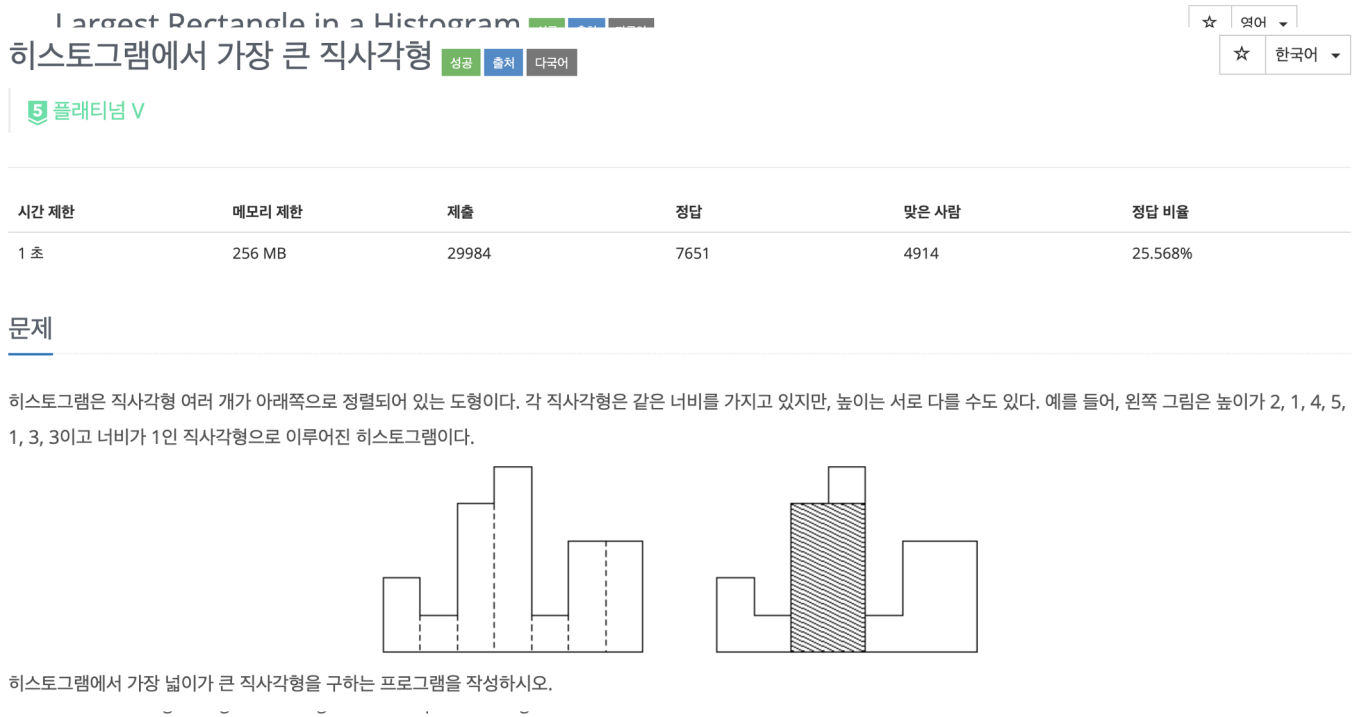
안태진(taejin7824@gmail.com)
GitHub(github.com/taejin1221)
상명대학교 소프트웨어학과
201821002

Contents

- Problem
- Approach
  - Segment Tree Review
  - Divide and Conquer Approach
- Code

Problem

- Problem
  - 문제 설명
    - 영어는 어려워...
  - 간단히 히스토그램에서 가장 큰 직사각형의 크기를 구하는 문제
  - DnC(Segment Tree), Stack 으로 풀 수 있음



Largest Rectangle in a Histogram

히스토그램에서 가장 큰 직사각형 성공 출처 다국어

플래티넘 V

☆ 영어

☆ 한국어

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	29984	7651	4914	25.568%

문제

히스토그램은 직사각형 여러 개가 아래쪽으로 정렬되어 있는 도형이다. 각 직사각형은 같은 너비를 가지고 있지만, 높이는 서로 다를 수도 있다. 예를 들어, 왼쪽 그림은 높이가 2, 1, 4, 5, 1, 3, 3이고 너비가 1인 직사각형으로 이루어진 히스토그램이다.

히스토그램에서 가장 넓이가 큰 직사각형을 구하는 프로그램을 작성하시오.

Approach

- Segment Tree Review
  - 특정 구간의 합(곱, 최솟값, 최댓값)을 $O(\log n)$만에 구할 수 있는 자료구조
  - 본 문제는 최솟값을 구하는데 Segment Tree를 사용
  - 재귀로 구현!
    - 오늘은 재귀 파티다! 오예

Approach

- Divide and Conquer Approach
  - 가장 작은 높이의 히스토그램을 선택

  1. 그 왼쪽의 넓이를 구함
  2. 오른쪽의 넓이 중 큰 것을 구함
  3. 자기 자신을 포함한 높이
     - 자신이 가장 작은 높이이기 때문에 왼쪽, 오른쪽의 길이 * 자신의 높이가 넓이

  - 위의 셋 중 가장 큰 것이 높이

Code

- Header file include 몇몇 define

```cpp
#include <iostream>

#define INF 1'234'567'890

using namespace std;

typedef long long ll;
```

- 몇몇 구조체 선언
  - Node: Segment Tree의 Node 구조체
    - 히스토그램의 높이와 배열에서의 index
    - min: Node 두개를 받아서 높이가 더 작은 Node를 반환

```cpp
Node min(const Node& n1, const Node& n2) {
    return (n1.height < n2.height) ? n1 : n2;
}
```

  - Range: 범위를 나타내는 class
    - getMid(): 중간값을 구하는 method
    - getDistance() 범위의 길이를 구하는 method

```cpp
struct Node {
    int height, idx;

    Node(): height(INF), idx(0) { }
    Node(int v, int i): height(v), idx(i) { }
};

class Range {
public:
    int start, end;

    Range(): start(0), end(0) { }
    Range(int s, int e): start(s), end(e) { }

    int getMid() {
        return (start + end) / 2;
    }

    int getDistance() {
        return (end - start) + 1;
    }
};
```

Code

- Segment Tree Class
  - target 구간 중 가장 작은 높이를 가지고 있는 값을 반환
  - 값은 {height, index}로 높이와 인덱스를 가지고 있음
  - 이외는 그냥 일반적인 최솟값 구하는 Segment Tree

```
// Min Segment Tree
class SegmentTree {
private:
    Node* tree;
    Node trash;
    int minIdx, maxIdx;

public:
    SegmentTree(int size, int arr[]) {
        tree = new Node[size * 4];
        minIdx = 0, maxIdx = size - 1;

        makeTree(1, Range(minIdx, maxIdx), arr);
    }

    ~SegmentTree() {
        delete[] tree;
    }
```

```
    void makeTree(int treeIdx, Range curr, int arr[]) {
        if (curr.start == curr.end)
            tree[treeIdx] = Node(arr[curr.start], curr.start);
        else {
            int mid = curr.getMid();
            int leftIdx = treeIdx * 2, rightIdx = treeIdx * 2 + 1;
            makeTree(leftIdx, Range(curr.start, mid), arr);
            makeTree(rightIdx, Range(mid + 1, curr.end), arr);

            tree[treeIdx] = min(tree[leftIdx], tree[rightIdx]);
        }
    }

    Node getMin(Range target) {
        return getMin(1, Range(minIdx, maxIdx), target);
    }

    Node getMin(int treeIdx, Range curr, Range target) {
        if (target.start <= curr.start && curr.end <= target.end)
            return tree[treeIdx];
        else if (curr.end < target.start || target.end < curr.start)
            return trash;
        else {
            return min(
                getMin(treeIdx * 2, Range(curr.start, curr.getMid()), target),
                getMin(treeIdx * 2 + 1, Range(curr.getMid() + 1, curr.end), target)
            );
        }
    }
};
```

Code

- 함수와 전역 변수 선언
  - solvedTestCase(): n을 전달 받아서 testcase를 푸는 함수

  - getLargestHistogram(Range curr)
    - 히스토그램 내 가장 큰 직사각형의 넓이를 구하는 함수

    - 분할정복으로 재귀적으로 값을 구함
      - 왼쪽, 오른쪽 중에 큰 값과 자신을 포함한 넓이 중 가장 큰 값을 반환

```
int histograms[100'000];
SegmentTree* tree;

ll getLargestHistogram(Range curr) {
    if (curr.start == curr.end)
        return histograms[curr.start];
    else if (curr.end < curr.start)
        return 0ll;

    Node minNode = tree->getMin(curr);

    ll result = max(
        getLargestHistogram(Range(curr.start, minNode.idx - 1)),
        getLargestHistogram(Range(minNode.idx + 1, curr.end))
    );

    return max(result, (long long)curr.getDistance() * minNode.height);
}

void solvedTestCase(int n) {
    for (int i = 0; i < n; i++)
        cin >> histograms[i];

    tree = new SegmentTree(n, histograms);

    cout << getLargestHistogram(Range(0, n - 1)) << '\n';

    delete tree;
}
```

Code

- main function
  - 빠른 입출력...
  - 입력 받고 테스트 케이스 풀자

```
int main(void) {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);

    int n;
    cin >> n;
    while (n) {
        solvedTestCase(n);

        cin >> n;
    }

    return 0;
}
```

P.S.

- Q: 이 문제를 선정하신 이유는?
  - A: 워낙 유명한 문제이기도 하고 전부터 계속 풀어야지 풀어야지 생각하고 있었음
  - 결정적인 이유는 완료가 중간에 끊겨 있어서 보기 싫었음 ㅋㄷㅋㄷ

17	정수론 및 조합론	정수론과 조합론을 배워 봅시다.	완료	12
18	스택	스택을 구현하고 사용해 봅시다.	완료	6
19	큐, 덱	큐와 덱을 구현하고 사용해 봅시다.	완료	7
20	분할 정복	재귀를 응용하는 알고리즘, 분할 정복을 익혀 봅시다.	완료	10
21	이분 탐색	이분 탐색 알고리즘을 배워 봅시다.	완료	7
22	우선순위 큐	가장 작은/큰 원소를 뽑는 자료구조를 배워 봅시다.	완료	4

- Q: 이 문제를 풀고 깨달은 것은?
  - A: 분할 정복이 꼭 절반으로 나누어 푸는 것이 아니다!
  - 지금까지 분할 정복은 Merge Sort처럼 정확히 절반으로 나눠 푸는 줄만 알았다.
  - 하지만 Quick Sort도 분할 정복인데 정확히 절반으로 나누지 않지 않는가!

감사합니다!