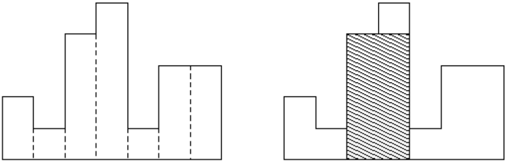
히스토그램에서 가장 큰 직사각형

2015년 10월 5일 baekjoon 댓글 (3개) 세그먼트 트리, Segment Tree, 스택, Stack, 히스토그램

6549번 문제: 히스토그램에서 가장 큰 직사각형

히스토그램에서 가장 큰 직사각형을 찾는 방법을 알아보겠습니다.

문제



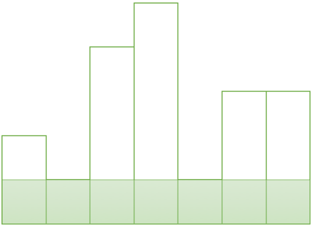
히스토그램에서 모든 막대의 너비는 1이고, 높이는 h_i일 때, 가장 넓이가 큰 직사각형을 찾아야 합니다.

모든 막대 x에 대해서, x를 높이로 하면서 만들 수 있는 가장 큰 직사각형을 찾아야 합니다.

분할 정복

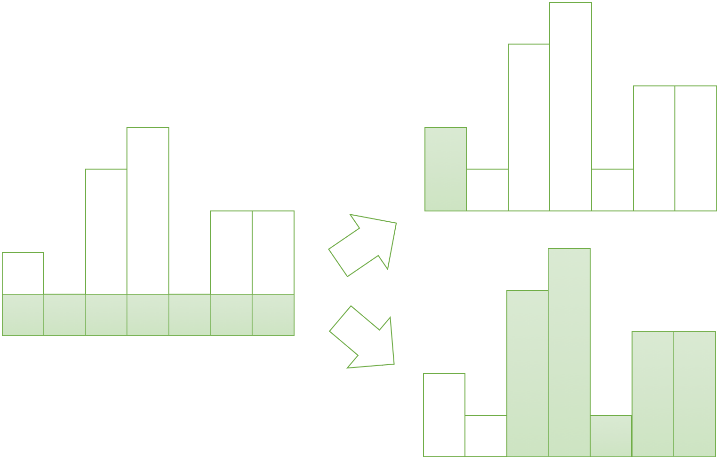
이 문제는 분할 정복으로 풀 수 있습니다.

어떤, 히스토그램이 있을 때, 그 히스토그램의 가장 왼쪽 끝과 오른쪽 끝을 변으로 하는 가장 큰 직사각형의 높이는 몇일까요? 높이는 히스토그램에서 가장 높이가 낮은 막대의 높이가 됩니다.

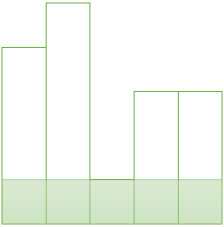


높이가 가장 낮은 막대의 번호를 m이라고 했을 때, 이 직사각형은 히스토그램에서 높이가 h_m이면서 만들 수

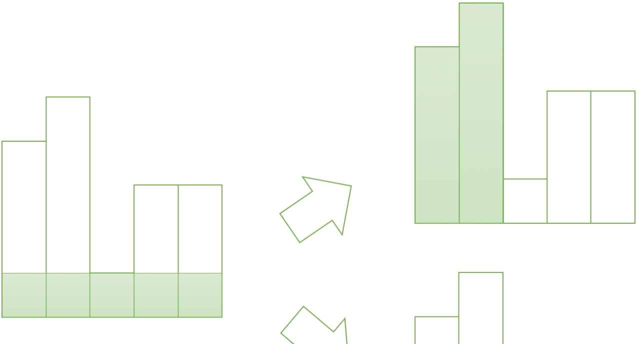
있는 가장 큰 직사각형이 됩니다. 그럼 이제, m의 왼쪽과 오른쪽으로 나눠서 문제를 풀 수 있습니다.

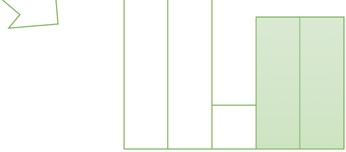


오른쪽 히스토그램에서 만들 수 있는 가장 큰 직사각형은 아래 그림과 같습니다.



그럼 여기서 또, 가장 높이가 낮은 막대 m을 찾고 왼쪽과 오른쪽으로 나눠서 문제를 풀 수 있습니다.





구간의 최소값은 블로그: 세그먼트 트리 (Segment Tree)를 이용해서 구할 수 있습니다. 따라서, 시간 복잡도는 O(NlgN)이 됩니다. (모든 막대에 대해서: O(N), 구간의 최소값을 구함: O(lgN))

구현할 때는, 세그먼트 트리에 최소값이 아닌, 최소값의 위치를 저장해놓아야 합니다.

```
#include <iostream>
#include <cmath>
#include <vector>
#include <algorithm>
using namespace std;
void init(vector<int> &a, vector<int> &tree, int node, int start, int end) {
    if (start == end) {
        tree[node] = start;
    } else {
        init(a, tree, node*2, start, (start+end)/2);
        init(a, tree, node*2+1, (start+end)/2+1, end);
        if (a[tree[node*2]] <= a[tree[node*2+1]]) {
            tree[node] = tree[node*2];
        } else {
            tree[node] = tree[node*2+1];
        }
    }
}
int query(vector<int> &a, vector<int> &tree, int node, int start, int end, int i, int
j) {
    if (i > end || j < start) {
        return -1;
    }
    if (i <= start && end <= j) {
        return tree[node];
    }
    int m1 = query(a, tree, 2*node, start, (start+end)/2, i, j);
    int m2 = query(a, tree, 2*node+1, (start+end)/2+1, end, i, j);
    if (m1 == -1) {
        return m2;
    } else if (m2 == -1) {
        return m1;
    } else {
        if (a[m1] <= a[m2]) {
            return m1;
        } else {
            return m2;
        }
    }
}
long long largest(vector<int> &a, vector<int> &tree, int start, int end) {
    int n = a.size();
    int m = query(a, tree, 1, 0, n-1, start, end);
    long long area = (long long)(end-start+1)*(long long)a[m];
    if (start <= m-1) {
        long long temp = largest(a, tree, start, m-1);
        if (area < temp) {
            area = temp;
        }
```

```
    }
    if (m+1 <= end) {
        long long temp = largest(a, tree, m+1, end);
        if (area < temp) {
            area = temp;
        }
    }
    return area;
}
int main() {
    while (true) {
        int n;
        cin >> n;
        if (n == 0) break;
        vector<int> a(n);
        for (int i=0; i<n; i++) {
            cin >> a[i];
        }
        int h = (int)(ceil(log2(n))+1e-9);
        int tree_size = (1 << (h+1));
        vector<int> tree(tree_size);
        init(a, tree, 1, 0, n-1);
        cout << largest(a, tree, 0, n-1) << '\n';
    }
    return 0;
}
```

스택

글의 첫 부분에도 써있지만, 정답을 구하려면 모든 막대 x에 대해서, x를 높이로 하면서 만들 수 있는 가장 큰 직사각형을 찾아야 합니다.

x를 높이로 하면서 만들 수 있는 가장 큰 직사각형을 찾기 위해서, x의 왼쪽에 있는 막대 중에 x보다 높이가 작은 첫 번째 막대 left와 오른쪽에 있는 막대 중에서 x보다 높이가 작은 첫 번째 막대 right를 찾아야 합니다.

스택에 막대를 하나씩 넣기 전에, 스택의 가장 위에 있는 막대 top과 현재 넣으려고 하는 막대 x를 비교해야 합니다. 만약, top의 높이가 x의 높이보다 크면, top을 높이로 하는 직사각형의 x를 지나갈 수 없습니다. top을 높이로하는 직사각형의 right는 x-1이 됩니다. top을 높이로 하는 직사각형의 left는 어디에 있을까요? 바로 top 다음에 스택에 들어있는 막대가 됩니다. 이제 left와 right를 구했기 때문에, top을 높이로 하는 직사각형의 넓이를 구할 수 있습니다.

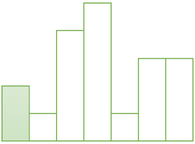
위의 과정이 모두 끝난 후에도 스택이 비어있지 않은 경우가 있습니다. 바로, right가 가장 오른쪽 끝인 경우입니다. 스택에 있는 막대를 하나씩 빼면서 위의 과정을 반복합니다. 이 때, right = n-1이 됩니다.

1. 0번 막대, 높이: 2

스택이 비어있기 때문에, 스택에 현재 막대의 번호 0을 넣습니다. 스택: `0`

2. 1번 막대, 높이 1

스택의 가장 위에 있는 막대보다 높이가 작습니다. 0번 막대의 오른쪽 끝은 1-1 = 0이 됩니다. 스택에서 pop을 수행하니 스택이 비어있게 됩니다. 왼쪽 끝은 0이 됩니다. 따라서, 0번 막대로 만들 수 있는 가장 큰 직사각형은 아래 그림과 같게 됩니다. 넓이: 2



이제 스택은 비어있게 되고, 스택에 현재 막대의 번호 1을 넣습니다. 스택: `1`

3. 2번 막대, 높이 4

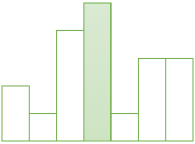
스택의 가장 위에 있는 1번 막대보다 높이가 크기 때문에, 스택에 현재 막대의 번호를 넣습니다. 스택: `1 2`

4. 3번 막대, 높이 5

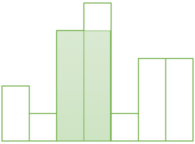
스택의 가장 위에 있는 2번 막대보다 높이가 크기 때문에, 스택에 현재 막대의 번호를 넣습니다. 스택: `1 2 3`

5. 4번 막대, 높이 1

스택의 가장 위에 있는 3번 막대의 높이 5가 현재 막대의 높이 1보다 큽니다. 스택에서 하나를 빼니 스택은 `1 2` 가 됩니다. right = 4-1 = 3, left = 2+1 = 3 이 됩니다. 3번 막대로 만들 수 있는 가장 큰 직사각형은 아래 그림과 같습니다. 넓이: 5



스택의 가장 위에 있는 2번 막대의 높이 4가 현재 막대의 높이 1보다 큽니다. 스택에서 하나를 빼니 스택은 `1` 가 됩니다. right = 4-1 = 3, left = 1+1 = 2 이 됩니다. 2번 막대로 만들 수 있는 가장 큰 직사각형은 아래 그림과 같습니다. 넓이: 8



스택의 가장 위에 있는 1번 막대보다 높이가 크지 않기 때문에, 스택에 현재 막대의 번호 4를 넣습니다. 스택: `1 4`

6. 5번 막대, 높이: 3

스택의 가장 위에 있는 4번 막대보다 높이가 크기 때문에, 스택에 현재 막대의 번호 5를 넣습니다. 스택: `1 4 5`

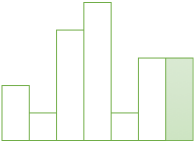
7. 6번 막대, 높이: 3

스택의 가장 위에 있는 4번 막대보다 높이가 크기 때문에, 스택에 현재 막대의 번호 5를 넣습니다. 스택: `1 4 5 6`

모든 막대가 스택에 들어갔고, 이제 right = n-1인 경우를 처리할 차례입니다.

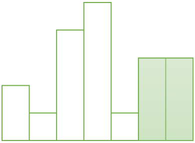
스택: `1 4 5 6`

막대 6으로 만들 수 있는 가장 큰 직사각형을 구해봅니다. 넓이: 3,



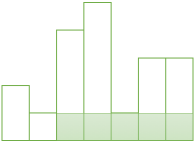
스택: `1 4 5`

막대 5으로 만들 수 있는 가장 큰 직사각형을 구해봅니다. 넓이: 6



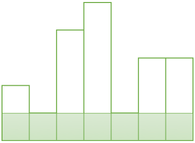
스택: `1 4`

막대 5으로 만들 수 있는 가장 큰 직사각형을 구해봅니다. 넓이: 5



스택: `1`

막대 1으로 만들 수 있는 가장 큰 직사각형을 구해봅니다. 넓이: 7



아래 소스는 6549번 문제: 히스토그램에서 가장 큰 직사각형와 같은 문제인 1725번 문제: 히스토그램를 푸는 소스입니다.

```
#include <cstdio>
#include <stack>
using namespace std;
int a[100000];
int main() {
    int n;
    scanf("%d",&n);
    for (int i=0; i<n; i++) {
        scanf("%d",&a[i]);
    }
    stack<int> s;
    int ans = 0;
    for (int i=0; i<n; i++) {
        int left = i;
        while(!s.empty() && a[s.top()] > a[i]) {
            int height = a[s.top()];
            s.pop();
            int width = i;
            if (!s.empty()) {
                width = (i - s.top() - 1);
            }
            if (ans < width*height) {
                ans = width*height;
            }
        }
        s.push(i);
    }
    while(!s.empty()) {
        int height = a[s.top()];
        s.pop();
        int width = n;
        if (!s.empty()) {
            width = n-s.top()-1;
        }
        if (ans < width*height) {
            ans = width*height;
        }
    }
    printf("%d\n",ans);
    return 0;
}
```

댓글 (3개)

댓글 쓰기

gkdlmc77

5년 전

오타났어요.

스택: 1 4 5

막대 5으로 만들 수 있는 가장 큰 직사각형을 구해봅니다. 넓이: 6

스택: 1 4

막대 5으로 만들 수 있는 가장 큰 직사각형을 구해봅니다. 넓이: 5

이렇게 막대 5로 만들 수 있는 가장 큰 직사각형을 두 번 썼어요.

스택: 1 4 막대 4로 만들 수 잇는 가장 큰 직사각형

---

gowithmylord 4년 전

감사합니다

---

lsn1106 11달 전

감사합니다

검색

목록보기

최신 글

알고리즘 문제풀이를 함에 있어서 자주 접하게 될 용어들

BOJ의 Good Bye, BOJ 2020!

BOJ의 2020년 마무리

2020-2021 ICPC 일정 정리

2020년 11월 21일 서버 사고

문제 출처 FAQ

채점하지 않는/못하는 언어

어떤 C, C++, Java, Python을 사용해야 할까?

2020년 7월 25일 서버 사고

11812 K진 트리 문제 공식의 수학적 증명

Baekjoon Online Judge

- 소개
- 뉴스
- 생중계
- 설문조사
- 블로그
- 캘린더
- 기부하기
- 기능 추가 요청
- 스페셜 저지 제작
- 실험실

채점 현황

- 채점 현황

문제

- 문제
- 단계별로 풀어보기
- 알고리즘 분류
- 새로 추가된 문제
- 새로 추가된 영어 문제
- 문제 순위
- 최근 제출된 문제
- 최근 풀린 문제
- 재채점 및 문제 수정

유저 대회 / 고등학교 대회

- FunctionCup kriiicon
- 구데기컵 꼬마컵 네블컵
- 소프트콘 웰노운컵
- HYEA Cup
- 경기과학고등학교
- 대구과학고등학교
- 부산일과학고
- 서울과학고등학교
- 선린인터넷고등학교

출처

- ICPC
- ICPC Korea Regional
- Olympiad
- 한국정보올림피아드
- 한국정보올림피아드시.도지역본선
- 전국 대학생 프로그래밍 대회 동아리 연합
- 대학교 대회
- 카카오 코드 페스티벌
- Coder's High

대학교 대회

- KAIST POSTECH
- 고려대학교 광주과학기술원
- 국민대학교 서강대학교
- 서울대학교 숭실대학교
- 아주대학교 연세대학교
- 인하대학교 전북대학교
- 중앙대학교 충남대학교
- 한양대 ERICA 홍익대학교
- 경인지역 6개대학 연합 프로그래밍 경시대회

도움말

- 채점 도움말 및 채점 환경
- 재채점 안내
- 런타임 에러 도움말
- 문제 스타일 안내
- 컴파일 또는 실행 옵션, 컴파일러 버전, 언어 도움말
- 문제집 도움말
- 대회 개최 안내
- 문제 출제 안내
- 이용 규칙
- 통계 도움말
- 질문 도움말
- 자주묻는 질문
- 강의 안내
- 짧은 주소 안내
- 광고 안내

© 2021 All Rights Reserved. 주식회사 스타트링크 | 서비스 약관 | 개인정보 보호 | 결제 이용 약관 | 도움말 | 광고 문의 | 업데이트 노트 | 이슈 | TODO

사업자 등록 번호: 541-88-00682
대표자명: 최백준
주소: 서울시 서초구 서초대로74길 29 서초파라곤 412호
전화번호: 02-521-0487 (이메일로 연락 주세요)
이메일: contacts@startlink.io
통신판매신고번호: 제 2017-서울서초-2193 호

STARTLINK

이 사이트는 ACM 또는 ICPC 대회와 무관하며, ACM으로부터 승인이나 지원을 받지 않고 있습니다.