Monotone Stack/Queue

나정휘

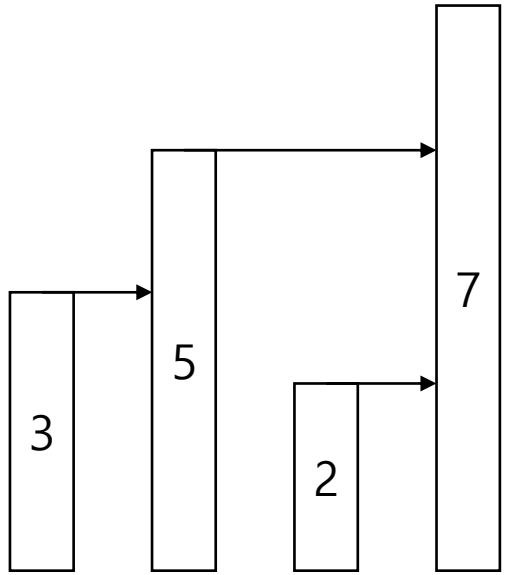
https://justiceHui.github.io/

목차

Monotone Stack

Monotone Stack (1)

- BOJ 17298 오큰수
  - 자신보다 오른쪽에 있으면서 자신보다 큰 수 중 가장 왼쪽에 있는 수를 구하는 문제
    - 3 5 2 7 → 5 7 7 -1
    - 막대 그래프로 그렸을 때, 각 막대의 시야를 차단하는 막대를 구하는 문제



Monotone Stack (1)

- BOJ 17290 오큰수
  - 스택을 사용한 풀이
    - 가장 오른쪽에 있는 원소부터 차례대로 보면서
    - 지금까지 본 원소 중 A[i]보다 큰 가장 왼쪽 원소를 구하면 됨

    - i < j이고 A[i] ≥ A[j]이면 A[j]는 더 이상 오큰수가 될 수 없음
      - A[i+1..N] 중 A[1..i]의 오큰수가 될 수 있는 후보는 오름차순
      - A[i]의 오큰수는 후보 중 A[i]보다 큰 첫 번째 원소
      - A[i]를 후보 집합에 넣을 때, A[i] 이하인 원소를 모두 제거해야 함

    - A[i]의 오큰수를 구하는 방법
      - 스택에서 A[i]보다 작거나 같은 수를 모두 제거한 뒤
      - 스택의 맨 위에 있는 원소가 오큰수
      - 오큰수를 구한 뒤 A[i]를 스택에 삽입

Monotone Stack (1)

- BOJ 17290 오큰수
  - 시간 복잡도: O(N)
    - 각 원소는 최대 한 번 스택에 들어가고
    - 최대 한 번 스택에서 빠져나옴
  - 이런 식으로 스택의 내용물의 단조성을 유지하는 것을 monotone stack 기법이라고 부름

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, A[1010101], B[1010101];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=1; i<=N; i++) cin >> A[i];

    stack<int> S;
    for(int i=N; i>=1; i--){
        while(!S.empty() && S.top() <= A[i]) S.pop();
        B[i] = S.empty() ? -1 : S.top();
        S.push(A[i]);
    }
    for(int i=1; i<=N; i++) cout << B[i] << " ";
}
```

질문?

Monotone Stack (2)

- BOJ 1725 히스토그램
  - 히스토그램에서 가장 큰 직사각형의 넓이를 구하는 문제
    - 분할 정복에서 했던 그 문제

  - 스택을 사용하면 O(N)에 해결할 수 있음
    - L[i] : A[i]의 왼쪽에 있으면서 A[i]보다 작으면서 가장 오른쪽에 있는 원소의 위치
    - R[i] : A[i]의 오른쪽에 있으면서 A[i]보다 작으면서 가장 왼쪽에 있는 원소의 위치

    - A[i]를 포함하면서 높이가 A[i]인 직사각형의 최대 너비 = R[i] – L[i] – 1
    - res = max(res, A[i] * (R[i] – L[i] – 1));

Monotone Stack (3)

- BOJ 5828 Fuel Economy
  - 문제 요약
    - 연료를 최대 G 만큼 담을 수 있는 연료 탱크
    - 1 만큼 이동할 때마다 1 만큼의 연료 소모
    - D 만큼 이동해야 함
    - N개의 주유소
      - 시작점으로부터 X[i] 만큼 떨어진 주유소
      - 연료 1 만큼 충전할 때마다 Y[i] 만큼의 비용이 필요
    - 연료가 B (≤ G) 만큼 있는 상태로 출발할 때
    - 목적지에 도달하기 위해 필요한 최소 연료

Monotone Stack (3)

- BOJ 5828 Fuel Economy
  - 만약 현재 주유소보다 싼 주유소에 갈 수 있다면
    - 그 주유소까지 갈 수 있을 만큼만 주유한 다음 이동하면 됨
    - 더 싼 주유소에서 주유하는 게 이득이기 때문
  - 만약 현재 주유소보다 싼 주유소가 없거나 너무 멀리 있다면(G 초과)
    - G 만큼 가득 채우고 다음 주유소로 이동
    - 앞으로 G 만큼 이동하는 동안 만나는 주유소는 더 비싸기 때문에 되도록 안 하는 게 이득

  - Next[i] = i보다 오른쪽에 있으면서 가격이 Y[i]보다 싼 가장 왼쪽에 있는 주유소
    - Next[i]까지의 거리가 G 이하라면 Next[i]로 이동
    - Next[i]까지의 거리가 G 초과라면 가득 채우고 다음 주유소로 이동

Monotone Stack (3)

- BOJ 5828 Fuel Economy
  - 주유소 정렬하는데 O(N log N)
  - 스택으로 Next 배열 구하는데 O(N)
  - 정답 구하는데 O(N)

  - 전체 시간 복잡도: O(N log N)

Monotone Stack (3)

- BOJ 5828 Fuel Economy

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, M, S, D; long long R; // n, max, now, dist, ans
pair<int,int> A[50505]; // pos, cost
int Next[50505];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M >> S >> D;
    for(int i=1; i<=N; i++) cin >> A[i].first >> A[i].second;
    A[++N].first = D; A[N].second = -1;
    sort(A+1, A+N+1);

    stack<pair<int,int>> stk; // cost, idx
    for(int i=N; i>=1; i--){
        while(!stk.empty() && stk.top().first >= A[i].second) stk.pop();
        Next[i] = stk.empty() ? -1 : stk.top().second;
        stk.emplace(A[i].second, i);
    }
    for(int i=1; i<=N; i++){
        S -= A[i].first - A[i-1].first;
        if(S < 0){ cout << -1; return 0; }
        int cost = min(M, A[Next[i]].first - A[i].first);
        R += 1LL * max(0, cost - S) * A[i].second;
        S = max(S, cost);
    }
    cout << R;
}
```

질문?

Monotone Queue

Monotone Queue (1)

- BOJ 11003 최솟값 찾기
  - 고정된 L에 대해, 모든 i에 대해 A[i-L+1..i]의 최솟값을 구하는 문제

  - 오큰수 문제에서 했던 것처럼, 최솟값이 될 수 있는 원소를 관리
    - 인덱스가 i-L+1보다 작은 원소는 더 이상 필요 없음
    - x < y이면서 A[x] > A[y]이면 x는 더 이상 최솟값이 될 수 없음
    - x < y이면서 A[x] = A[y]이면 y가 더 오래 살아있으므로 x는 신경 쓰지 않아도 됨

    - 오큰수 문제에서 인덱스가 i-L+1 미만인 원소를 제거하는 것만 추가됨
    - +) A[i]를 넣을 때 A[i] 이상인 원소 모두 제거

Monotone Queue (1)

- BOJ 11003 최솟값 찾기
  - A[i-L+1..i]의 최솟값을 구하는 과정
    - 인덱스가 i-L+1보다 작은 후보를 모두 제거
    - A[i]보다 크거나 같은 후보 모두 제거
    - A[i]를 후보에 삽입

    - deque로 관리하면 위 3가지 작업은 pop_front, pop_back, push_back으로 할 수 있음
    - deque의 내용물은 증가하는 상태
    - deque의 맨 앞에 있는 원소가 최솟값

Monotone Queue (1)

- BOJ 11003 최솟값 찾기
  - 시간 복잡도: O(N)
  - monotone stack과 비슷하게 monotone queue라고 부름

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    int N, L; cin >> N >> L;

    deque<pair<int,int>> dq;
    for(int i=1; i<=N; i++){
        int t; cin >> t;
        while(!dq.empty() && dq.front().second < i-L+1) dq.pop_front();
        while(!dq.empty() && dq.back().first >= t) dq.pop_back();
        dq.push_back(make_pair(t, i));
        cout << dq.front().first << " ";
    }
}
```

질문?

Monotone Queue (2)

- BOJ 15678 연세워터파크
  - DP[i] = max DP[i-k] + A[i] (1 ≤ k ≤ D)
  - 똑같죠?

  - 이런 식으로 자료구조를 사용해 점화식을 계산하는 건 4차시(DP2)에서 더 자세하게 다룰 예정
    - DP + Stack / Queue / Heap / Segment Tree / ...

Monotone Queue (3)

- BOJ 3988 수 고르기
  - 문제 요약
    - N개의 수로 구성된 수열에서 원소 K개 제거
    - (두 원소의 차이의 최댓값) + (두 원소의 차이의 최솟값)을 최소화
  - 관찰
    - 정렬된 수열을 가정하자.
    - 두 원소의 차이의 최댓값 = 수열의 최댓값 – 수열의 최솟값
    - 두 원소의 차이의 최솟값 = 인접한 두 수의 차이의 최솟값
    - 차이의 최솟값을 최소화하기 위해서는 인접한 수들을 선택해야 함
    - 차이의 최댓값을 최소화하기 위해서는 인접한 N-K개의 수를 선택해야 함

Monotone Queue (3)

- BOJ 3988 수 고르기
  - 모든 $1 \le i \le N-K+1$ 에 대해 $A[i+K-1] - A[i] + \min(A[j+1] - A[j])$의 최솟값을 구하면 됨
    - $i \le j < i+K-1$
  - BOJ 11003 최솟값 찾기의 풀이를 그대로 사용하면 됨

질문?

과제

- 필수
  - 17298 오큰수
  - 1725 히스토그램
  - 5828 Fuel Economy
  - 11003 최솟값 찾기
  - 15678 연세워터파크
  - 3988 수 고르기
- 심화
  - 2867 수열의 값
  - 11873 최대 직사각형
  - 8201 Pilots
  - 14869 요리 강좌