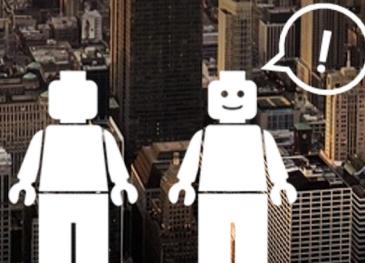


# Monotone Stack & Queue

2111618 길수민



# INDEX

## 01 Monotone Stack

- 핵심 아이디어
- 활용 방법

## 02 Monotone Queue

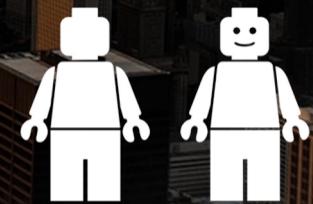
- 핵심 아이디어
- 활용 방법

## 03 Practice

- Monotone Stack
- Monotone Queue

## 04 Assignment

- 문제 설명
- 약간의 힌트





# 1. Monotone Stack

핵심 아이디어  
활용 방법

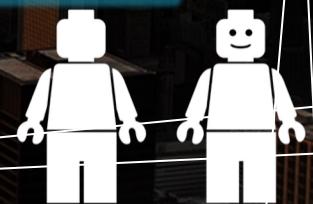
# 1. Monotone Stack

Monotone Stack: 몇몇 문제들의 시간 복잡도를  
 $O(N)$  정도로 줄여주는 강력한 테크닉

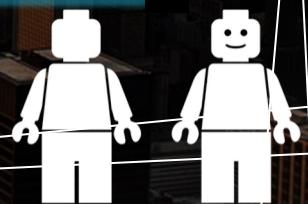
# 1. Monotone Stack

핵심 아이디어 : 스택의 원소들을 단조롭게 유지

-> 오름차순 혹은 내림차순



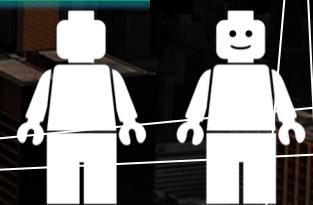
# 1. Monotone Stack



# 1. Monotone Stack



5



# 1. Monotone Stack

9

2

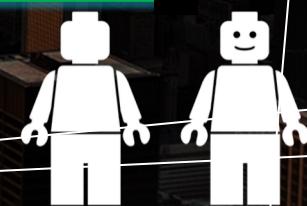
5

11

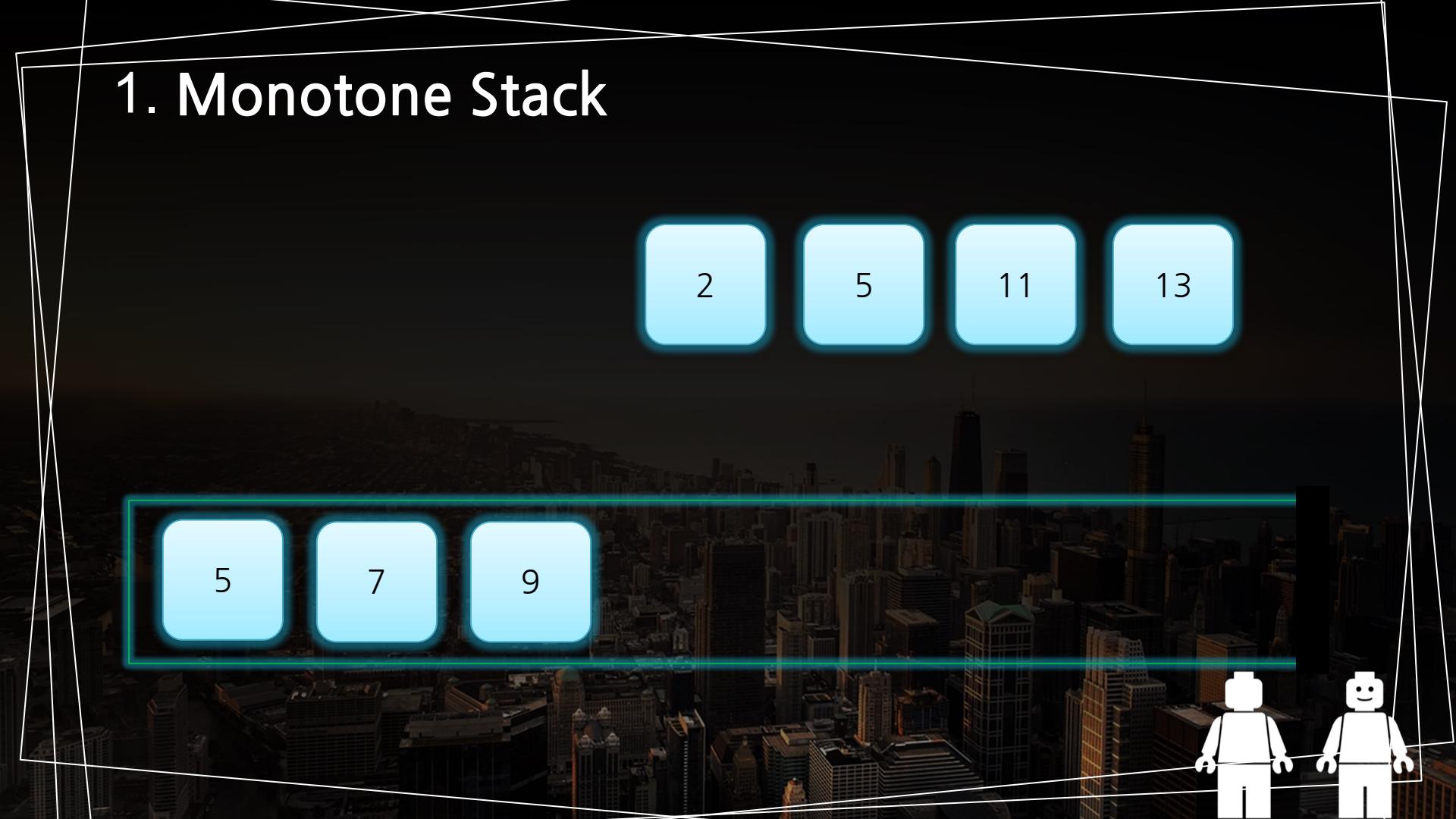
13

5

7



# 1. Monotone Stack



2    5    11    13

5    7    9

# 1. Monotone Stack

5

11

13

2



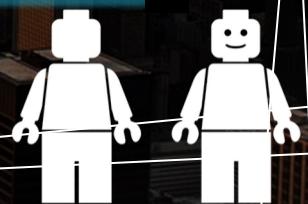
# 1. Monotone Stack

2

5

11

13



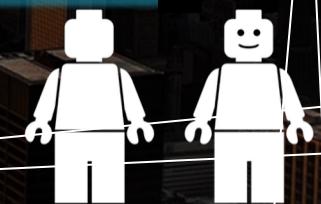
# 1. Monotone Stack

13

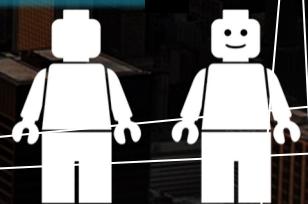
2

5

11



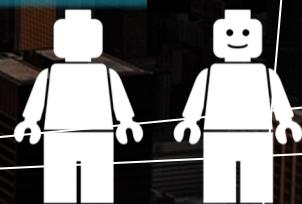
# 1. Monotone Stack



# 1. Monotone Stack



5 이상 X



# 1. Monotone Stack

5

7

9

2

5

11

13

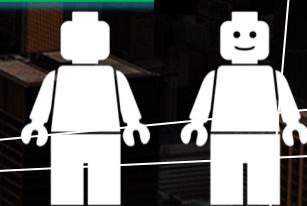
11 이상 X

2

5

11

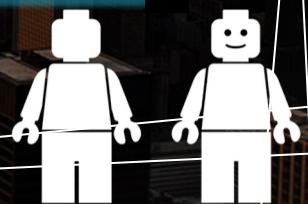
13



# 1. Monotone Stack



13 이상 X

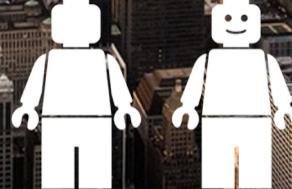




## 2. Monotone Queue

핵심 아이디어

활용 방법



## 2. Monotone Queue

Monotone Queue : Sliding Window에서  
min/max 값을 찾을 때 유리한 테크닉

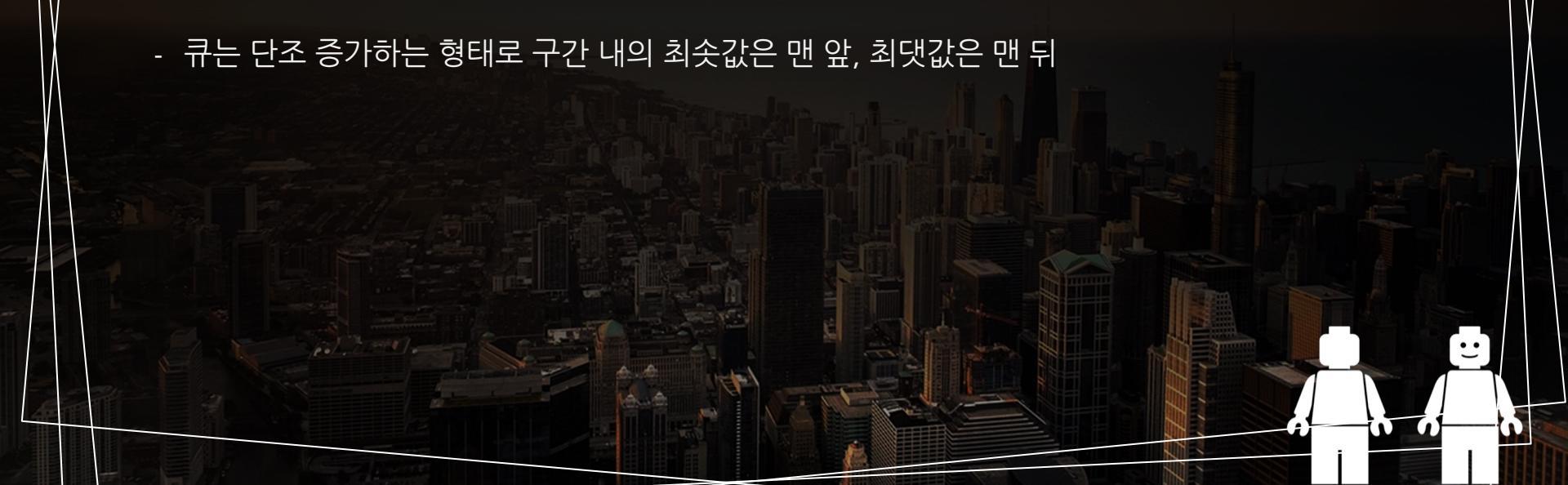
## 2. Monotone Queue

핵심 아이디어 : 일정 범위 내의 ‘유효한’ 값만

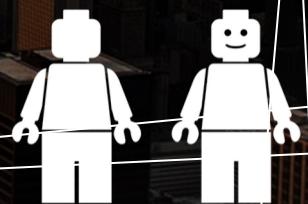
저장하고 heap 처럼 맨 앞에 min/max값 저장

## 2. Monotone Queue

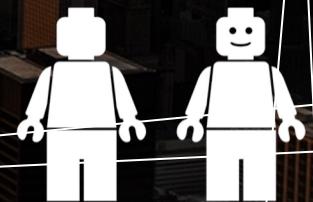
- 모노톤 큐의 맨 앞부분을 보면서 현재 위치까지 영향력을 줄 수 없는 모든 원소들을 앞에서 제거
- 모노톤 큐의 뒷 부분을 보면서 현재 넣고자 하는 값보다 큰 원소들 모두 제거
- 현재 위치의 값 삽입
- 큐는 단조 증가하는 형태로 구간 내의 최솟값은 맨 앞, 최댓값은 맨 뒤



## 2. Monotone Queue



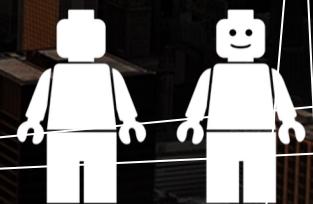
## 2. Monotone Queue



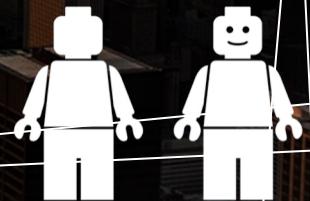
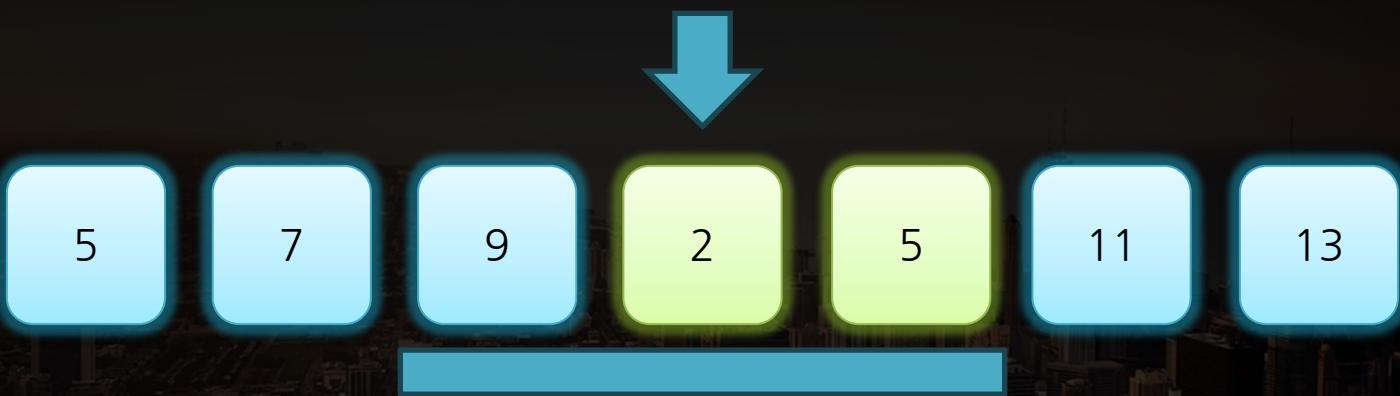
## 2. Monotone Queue



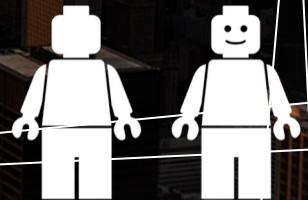
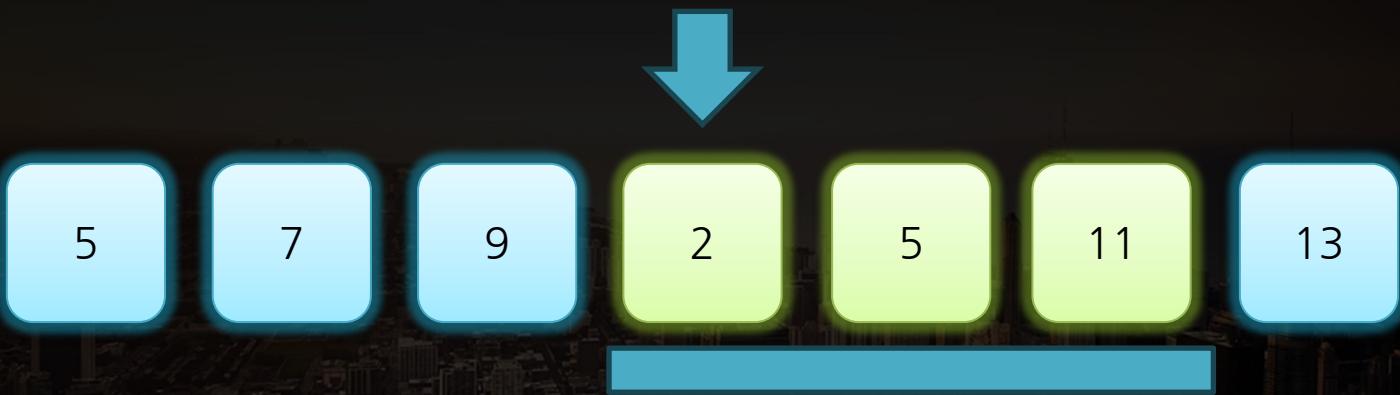
## 2. Monotone Queue



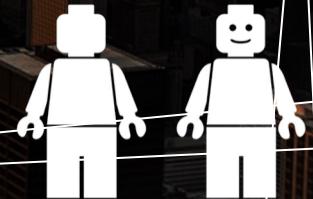
## 2. Monotone Queue



## 2. Monotone Queue



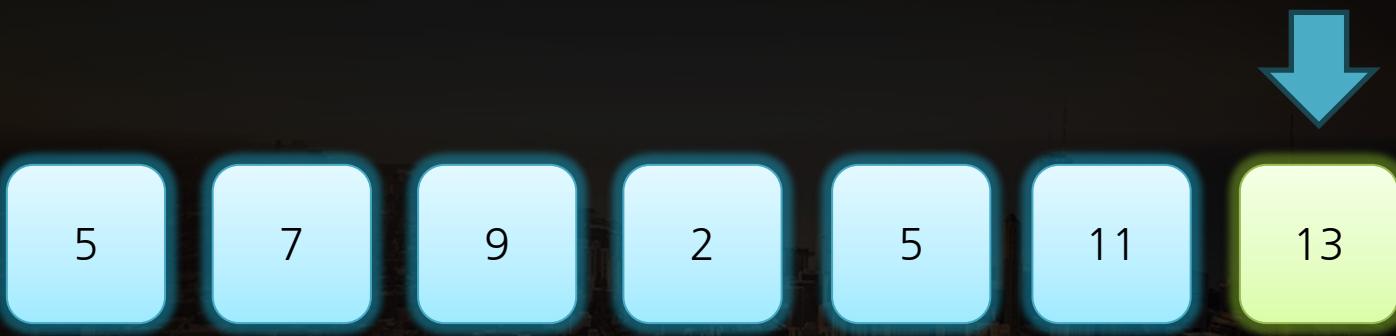
## 2. Monotone Queue

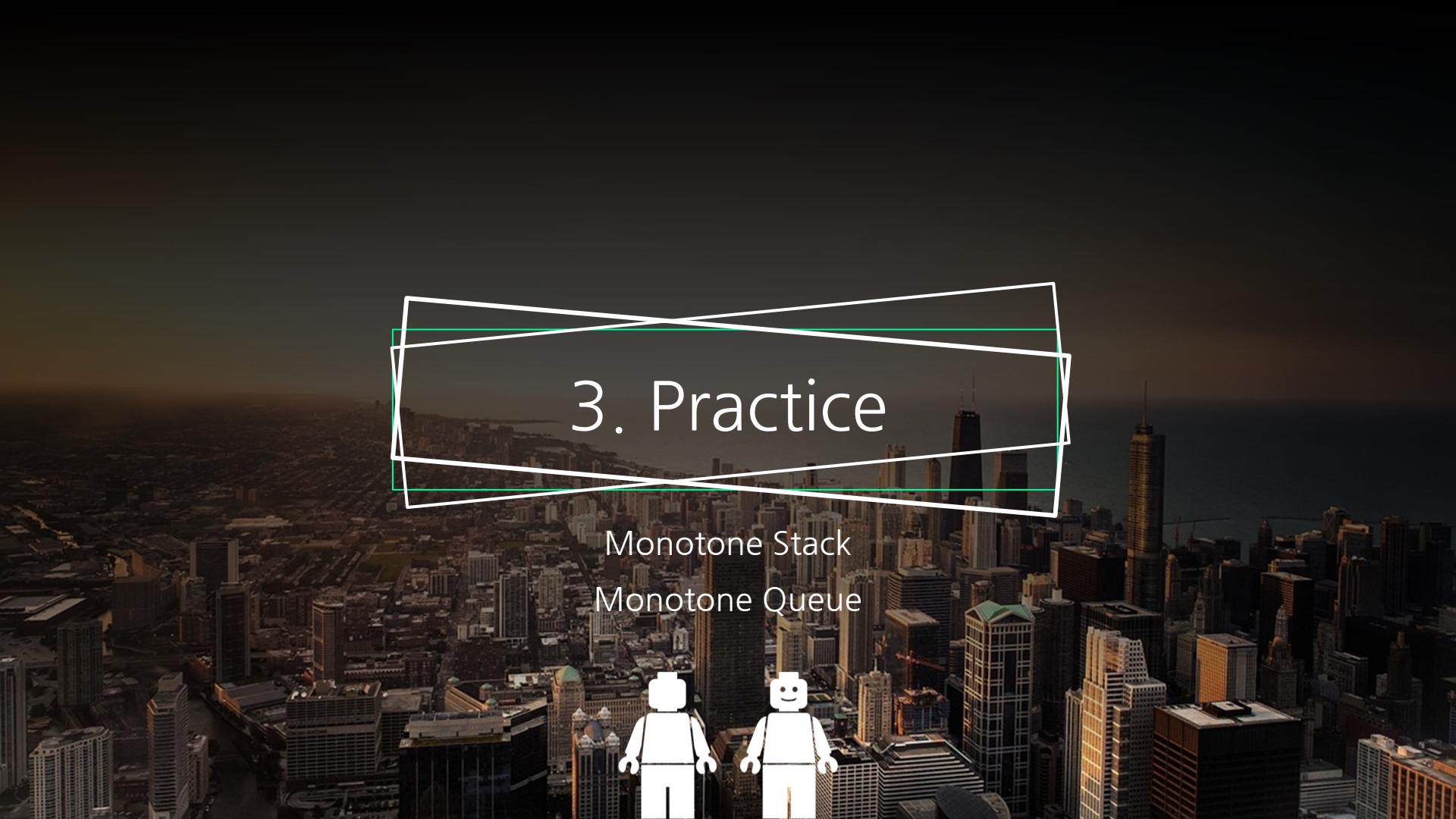


## 2. Monotone Queue



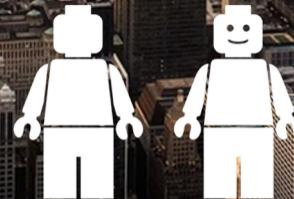
## 2. Monotone Queue





### 3. Practice

Monotone Stack  
Monotone Queue



# 3. Practice

## BOJ 17298 오큰수

17298번 제출 맞힌 사람 솔코딩 재채점 결과 체점 :

**오큰수** 성공  
4 골드 IV

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	512 MB	74498	26093	18437	33.604%

**문제**

크기가 N인 수열  $A = A_1, A_2, \dots, A_N$ 이 있다. 수열의 각 원소  $A_i$ 에 대해서 오른수  $NGE(i)$ 를 구하려고 한다.  $A_i$ 의 오른쪽에 있으면서  $A_i$ 보다 큰 수 중에서 가장 왼쪽에 있는 수를 의미한다. 그러한 수가 없는 경우에 오른수는 -1이다.

예를 들어,  $A = [3, 5, 2, 7]$ 인 경우  $NGE(1) = 5$ ,  $NGE(2) = 7$ ,  $NGE(3) = 7$ ,  $NGE(4) = -1$ 이다.  $A = [9, 5, 4, 8]$ 인 경우에는  $NGE(1) = -1$ ,  $NGE(2) = 8$ ,  $NGE(3) = 8$ ,  $NGE(4) = -1$ 이다.

**입력**

첫째 줄에 수열 A의 크기 N ( $1 \leq N \leq 1,000,000$ )이 주어진다. 둘째 줄에 수열 A의 원소  $A_1, A_2, \dots, A_N$  ( $1 \leq A_i \leq 1,000,000$ )이 주어진다.

**출력**

총 N개의 수  $NGE(1), NGE(2), \dots, NGE(N)$ 을 공백으로 구분해 출력한다.

예제 입력 1 복사

```
4
3 5 2 7
```

예제 출력 1 복사

```
5 7 7 -1
```

예제 입력 2 복사

```
4
9 5 4 8
```

예제 출력 2 복사

```
-1 8 8 -1
```



### 3. Practice

BOJ 17298 오큰수

- 자신보다 오른쪽에 있으면서 자신보다 큰 수 중 가장 왼쪽 수를 구하는 문제
- 가장 오른쪽에 있는 원소부터 차례대로
- 지금까지 본 원소 중  $A[i]$ 보다 큰 가장 왼쪽 원소를 구하기
- 스택에서  $A[i]$ 보다 작거나 같은 수를 모두 제거한 뒤
- 스택의 맨 위에 있는 원소가 오큰수
- 스택에  $A[i]$  삽입

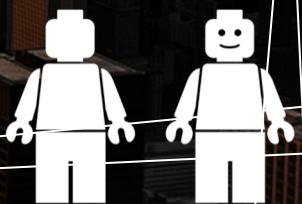
# 3. Practice

```
#include <bits/stdc++.h>

using namespace std;

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

    int N;
    cin >> N;
    vector<int> arr(N);
    for (int i = 0; i < N; i++) cin >> arr[i];
    stack<int> st;
    vector<int> ans;
    for (int i = N - 1; i >= 0; i--) {
        while (!st.empty() && st.top() <= arr[i])
            st.pop();
        if (st.empty())
            ans.push_back (-1);
        else ans.push_back (st.top());
        st.push (arr[i]);
    }
    reverse (ans.begin(), ans.end());
    for (auto& it : ans) cout << it << " ";
    cout << "\n";
    return 0;
}
```



# 3. Practice

## BOJ 1725 히스토그램

### 입력

첫 행에는  $N$  ( $1 \leq N \leq 100,000$ ) 이 주어진다.  $N$ 은 히스토그램의 가로 칸의 수이다. 다음  $N$  행에 걸쳐 각 칸의 높이가 원쪽에서부터 차례대로 주어진다. 각 칸의 높이는 1,000,000,000보다 작거나 같은 자연수 또는 0이다.

### 출력

첫째 줄에 가장 큰 직사각형의 넓이를 출력한다. 이 값은 20억을 넘지 않는다.

### 예제 입력 1

복사

7  
2  
1  
4  
5  
1  
3  
3

### 예제 출력 1

복사

8

5 1725번 제출 맞힌 사람 속코딩 재체점 결과 체점 현황 내 제출 난이도 기여 ☰ 질문 게시판

히스토그램 성공 다국어 ☆ 한국어 ▾

5 플래티넘 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
0.7 초	128 MB	23517	8732	6224	39.686%

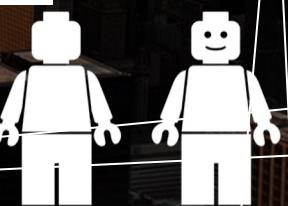
문제

히스토그램에 대해서 알고 있는가? 히스토그램은 아래와 같은 막대그래프를 말한다.

각 칸의 간격은 일정하고, 높이는 어떤 정수로 주어진다. 위 그림의 경우 높이가 각각 2 1 4 5 1 3 3이다.

이러한 히스토그램의 내부에 가장 넓이가 큰 직사각형을 그리려고 한다. 아래 그림의 빛금 칸 부분이 그 예이다. 이 직사각형의 밑변은 항상 히스토그램의 아랫면에 평행하게 그려져야 한다.

주어진 히스토그램에 대해, 가장 큰 직사각형의 넓이를 구하는 프로그램을 작성하시오.



### 3. Practice

BOJ 1725 히스토그램

- 스택을 활용하여  $O(N)$ 에 해결 가능
- $L[i]$  :  $A[i]$ 의 왼쪽에 있는  $A[i]$ 보다 작은 가장 오른쪽 원소의 위치
- $R[i]$  :  $A[i]$ 의 오른쪽에 있는  $A[i]$ 보다 작은 가장 왼쪽 원소의 위치
- 원소  $A[i]$ 를 기준으로 한 넓이 :  $(R[i] - L[i] - 1) * A[i]$

# 3. Practice

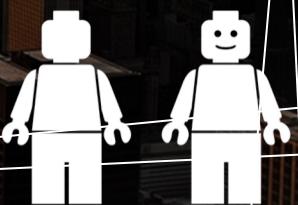
```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

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

    int N;
    cin >> N;
    vector<int> arr(N);
    vector<int> L(N, 0), R(N, 0);
    for (int i = 0; i < N; i++) cin >> arr[i];
    stack<int> stkL, stkR;
    for (int i = 0; i < N; i++) {
        while (!stkL.empty() && arr[stkL.top()] >= arr[i]) stkL.pop();
        L[i] = stkL.empty() ? -1 : stkL.top();
        stkL.push (i);
    }
    for (int i = N - 1; i >= 0; i--) {
        while (!stkR.empty() && arr[stkR.top()] >= arr[i]) stkR.pop();
        R[i] = stkR.empty() ? N : stkR.top();
        stkR.push (i);
    }
    ll ans = 0;
    for (int i = 0; i < N; i++) {
        ll sub = 1LL * (R[i] - L[i] - 1) * arr[i];
        ans = max (ans, sub);
    }
    cout << ans << "\n";
}

return 0;
}
```



# 3. Practice

## BOJ 11003 최솟값 찾기

11003번 제출 맞힌 사람 솔코딩 재제출 결과 채점 현:

최솟값 찾기 성과

5 플래티넘 V

시간 제한	메모리 제한	제출	정답	맞힌 사람
2.4 초 (하단 참고)	512 MB	31897	9672	6238

문제

N개의 수  $A_1, A_2, \dots, A_N$ 과 L이 주어진다.

$D_i = A_{i:L+1} \sim A_i$  중의 최솟값이라고 할 때, D에 저장된 수를 출력하는 프로그램을 작성하시오. 이때,  $i \leq 0$  인  $A_i$ 는 무시하고 D를 구해야 한다.

입력

첫째 줄에 N과 L이 주어진다. ( $1 \leq L \leq N \leq 5,000,000$ )

둘째 줄에는 N개의 수  $A_i$ 가 주어진다. ( $-10^9 \leq A_i \leq 10^9$ )

출력

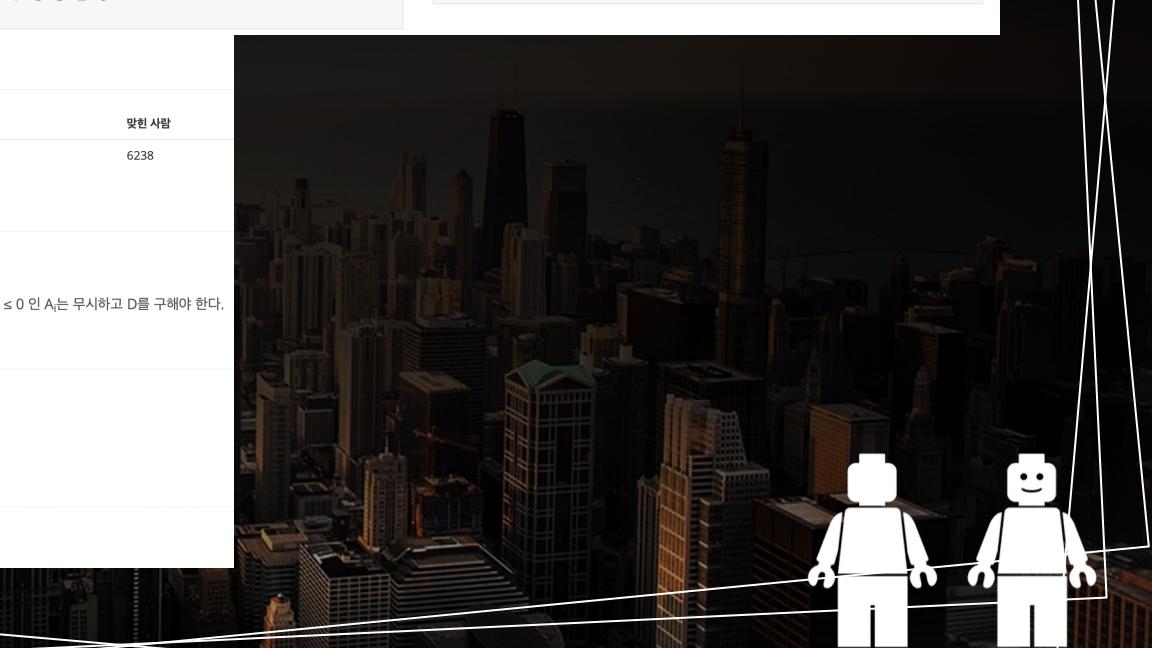
첫째 줄에  $D_i$ 를 공백으로 구분하여 순서대로 출력한다.

예제 입력 1 복사

12 3  
1 5 2 3 6 2 3 7 3 5 2 6

예제 출력 1 복사

1 1 1 2 2 2 2 2 3 3 3 2 2



### 3. Practice

BOJ 11003 최솟값 찾기

- Monotone Stack과 같이 최솟값이 될 수 있는 원소들을 관리
- 슬라이딩 윈도우처럼 진행하고 해당 구간에 속하지 않는 원소들을 제거
- $A[i]$ 보다 크거나 같은 원소들 모두 제거 (해당 문제에서 같은 건 신경쓰지 않아도 괜찮음)
- $A[i]$ 를 deque에 삽입
- 원소가 증가하는 상태로 deque로 관리 (`pop_front`, `pop_back`, `push_back`)
- Deque의 맨 앞에 있는 원소가 최솟값

# 3. Practice

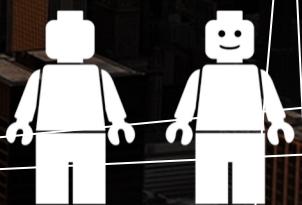


```
#include <bits/stdc++.h>
#define X first
#define Y second

using namespace std;
typedef pair<int, int> pi;

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

    int N, L;
    cin >> N >> L;
    deque<pi> dq;
    for (int i = 0; i < N; i++) {
        int t;
        cin >> t;
        while (!dq.empty() && dq.front().Y < i - L + 1) dq.pop_front();
        while (!dq.empty() && dq.back().X >= t) dq.pop_back();
        dq.push_back ({t, i});
        cout << dq.front().X << " ";
    }
    cout << "\n";
    return 0;
}
```

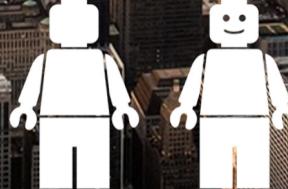




# 4. Assignment

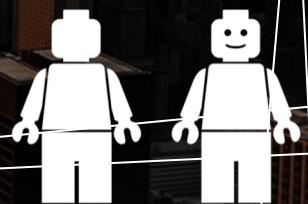
문제 설명

약간의 힌트



# 4. Assignment

- 6198 옥상 정원 꾸미기 (G5)
- 5828 Fuel Economy (P5)
- 2867 수열의 값 (P4)
- 15678 연세워터파크 (P5)
- 3988 수 고르기 (P4)



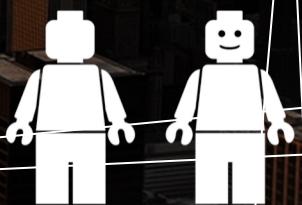
# 4. Assignment

## 5828 Fuel Economy (P5)

- 연료 최대 G만큼 담을 수 있는 연료 탱크
- 1만큼 이동할 때마다 1만큼 연료 소모, 목표 : D만큼 이동
- i번째 주유소는  $X[i]$ 만큼 떨어져 있고 연료 1의 충전 비용은  $Y[i]$  (주유소 N개)
- 연료가 B만큼 있는 상태로 출발 ( $B \leq G$ )
- 목적지에 도달하기 위해 필요한 최소 연료 구하기

## 2867 수열의 합

- (최댓값의 합) - (최솟값의 합)  $\rightarrow$  최댓값 단조 증가, 최솟값 단조 감소
- DP 활용 : 각 인덱스를 마지막 원소로 하는 부분 수열에서의 최대/최솟값



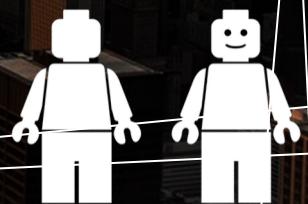
# 4. Assignment

15678 연세워터파크 (P5)

- DP 이용
- $DP[i] = \max(DP[i - k]) + A[i] \quad (1 \leq k \leq D)$

- 3988 수 고르기 (P4)

- K개의 원소를 제거한 배열에서 두 원소 차이의 최댓값과 최솟값의 합을 최소화하는 문제
- 정렬된 배열에서 인접한 수를 선택해야 차이의 최솟값을 최소화할 수 있음
- N - K개의 수를 포함하는 각 구간에 대해서 모두 값 계산





# Q&A





# THANK YOU

