ICPC Sinchon

2022 Winter Algorithm Camp

5. 선형 자료구조

서강대학교 임지환

2022 Winter Algorithm Camp

목차

Introduction – 동적 배열

* 동적 배열 (Dynamic Array)

- 메모리의 연속된 위치에 저장
- 주어진 위치의 원소에 접근하거나 변경하는 동작을 $O(1)$에 수행
- 배열 크기를 변경할 수 있음

* list append at python, vector push_back at C++

- 메모리에 연속적으로 저장되면 무한정 늘릴 수는 없지 않을까? -> reallocation, $O(N)$
- 상수시간 ($O(1)$) 아닌가요?

2022 Winter Algorithm Camp

Introduction –메모리 연속성과 수행 시간

* 메모리 연속성 & 지역성 (Locality)

- 1차원 배열뿐만 아니라 다차원 배열도 연속된 공간에 데이터가 저장된다.

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

a[0][0]
a[0][1]
…
a[1][0]
…
a[2][0]
…

- Temporal locality : 가장 최근에 읽어온 data는 다시 읽어올 때도 빠르게 access한다.
- Spatial locality : 프로그램 수행 단계에서 접근하는 메모리의 영역은 이미 접근이 이루어진 영역의 근처일 가능성이 높다.

2022 Winter Algorithm Camp

Introduction –메모리 연속성과 수행 시간

* 무엇이 더 빠를까?

```
1 int a[10000][10000];
2
3 for (int i = 0; i < 10000; i++)
4     for (int j = 0; j < 10000; j++)
5         a[i][j] += i + j;
```

```
1 int a[10000][10000];
2
3 for (int j = 0; j < 10000; j++)
4     for (int i = 0; i < 10000; i++)
5         a[i][j] += i + j;
```

2022 Winter Algorithm Camp

Stack – Introduction

* 소개

- 후입 선출 (LIFO; Last in First Out)
- 선입 후출 (FILO; First In Last Out)
- Underflow & Overflow

* Basic operations (simulation)

- Push: stack의 Top 위에 원소 추가
- Pop: 가장 마지막에 넣은 원소 제거
- Top(or Peek): 가장 마지막에 넣은 원소 반환
- isEmpty: stack이 비어 있으면 True, 아니면 False

2022 Winter Algorithm Camp

9012. 괄호

길이가 50이하인 '('과 ')'만으로 이루어진 문자열

주어진 문자열은 올바른 괄호 문자열인가?

2022 Winter Algorithm Camp

9012. 괄호

1. 올바른 괄호문자열 (Valid PS)의 특징을 먼저 알아보자.

1) 빈 문자열은 Valid PS이다.
2) A가 Valid PS이면 (A)도 Valid PS이다.
3) A와 B가 Valid PS이면 AB도 Valid PS이다.

2022 Winter Algorithm Camp

9012. 괄호

1. 올바른 괄호문자열 (Valid PS)의 특징을 먼저 알아보자.

1) 빈 문자열은 Valid PS이다. -> Valid PS인 경우 빈 문자열로 고려해도 되겠다.
2) A가 Valid PS이면 (A)도 Valid PS이다.
3) A와 B가 Valid PS이면 AB도 Valid PS이다.

2022 Winter Algorithm Camp

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

2022 Winter Algorithm Camp

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

2022 Winter Algorithm Camp

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

2022 Winter Algorithm Camp

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

2022 Winter Algorithm Camp

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

9012. 괄호

2. ex: “ (( ) ( )) (( )) ”

2022 Winter Algorithm Camp

2841. 외계인의 기타 연주

기타줄이 6개, 각 줄마다의 프렛 수는 30만 이하. 손가락은 무한개

길이 50만 이하의 멜로디가 (줄 번호, 프렛 번호) 형태로 주어짐

어떤 줄의 프렛을 여러 개 누르고 있다면 가장 높은 프렛의 음이 발생

ex: 3번 줄의 5번 프렛을 누르고 있는 상태에서
7번 프렛을 누른 음을 연주하려면 손가락을 뗄 필요가 없이 7번 누르고 연주
2번 프렛을 연주하려면 5번과 7번을 떼고 2를 누르고 연주

손가락으로 프렛을 한 번 누르거나 떼는 것을 손가락을 한 번 움직였다고 할 때, 손가락의 최소 움직임 횟수?

2022 Winter Algorithm Camp

2841. 외계인의 기타 연주

1. 누르기만 하면 되는 경우

해당 줄에서 **누르고 있던 가장 높은 프렛이 현재 내야 하는 음보다 낮을 때**는 현재 프렛이 기존 것들을 모두 커버하기 때문에 누르기만 하면 된다.

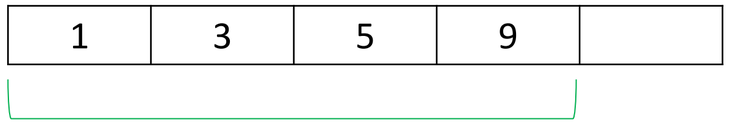
given 11:

2841. 외계인의 기타 연주

2. 떼고 눌러야 하는 경우

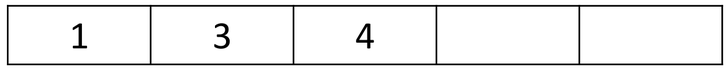
해당 줄에서 현재 내고자 하는 음보다 높은 프렛은 전부 떼야 한다.
-> 해당 줄에서 **누르고 있는 가장 높은 프렛보다 현재 내고자 하는 음이 높아지게끔** 해야 한다.

given 4:



기존에 누르고 있던 프렛

pop 9, 5



2841. 외계인의 기타 연주

3. 종합하면

1) 하나의 줄 내부에서 누른 프렛은 증가하는 형태로 관리되어야 한다.
2) 손가락을 떼기 위해서는 높은 것부터 순차적으로 떼야 한다.

각 줄에 눌린 프렛은 스택 형태로, 그 내부는 증가하게 관리(monotone stack)

* stack을 monotonic하게 관리하는 법(ex: monotone increasing)

```
monotonic (a[]):
    st // stack
    for i <- 0 to n
        while st is not empty and st's top >= y
            pop from st
        push a[i] to st
```

2022 Winter Algorithm Camp

Queue – introduction

* 소개

- 선입선출 (FIFO; First In First Out)
- Underflow & Overflow

* Basic operations (simulation)

- Enqueue: queue의 Rear 뒤에 원소 추가
- Dequeue: queue의 원소 중 가장 먼저 넣은 원소 제거
- Front: 가장 먼저 넣은 원소 반환
- Rear: 가장 마지막에 넣은 원소 반환
- isEmpty: queue가 비어 있으면 True, 아니면 False

2022 Winter Algorithm Camp

1158. 요세푸스 문제

1번 부터 N번까지 N명의 사람이 원을 이루면서 앉아 있음 $(1 \leq N \leq 5{,}000)$

순서대로 K번째 번째 사람을 제거하며 N명이 모두 제거될 때까지 계속됨 $(1 \leq K \leq 5{,}000)$

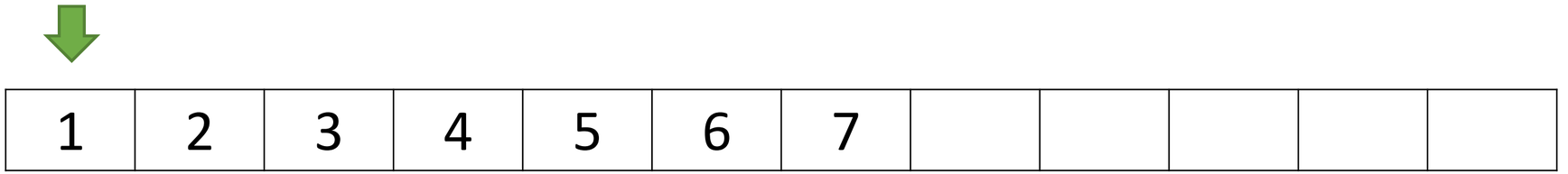
제거되는 순서를 구해보자.

2022 Winter Algorithm Camp

1158. 요세푸스 문제

1. K번째 사람 찾기 및 제거를 큐를 통해 구현해보자.

(N, K) = (7, 3)

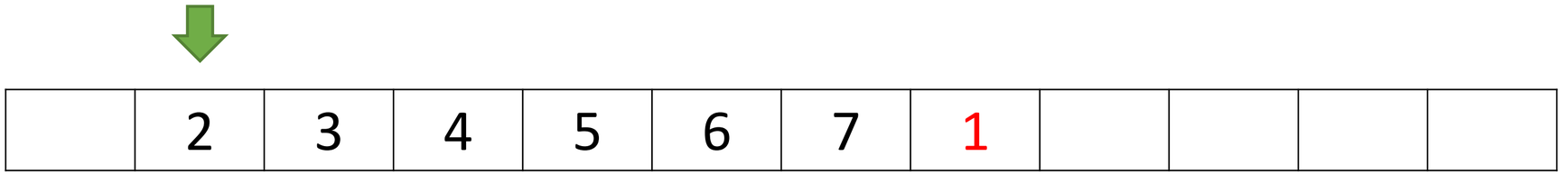


2022 Winter Algorithm Camp

1158. 요세푸스 문제

1. K번째 사람 찾기 및 제거를 큐를 통해 구현해보자.

(N, K) = (7, 3)

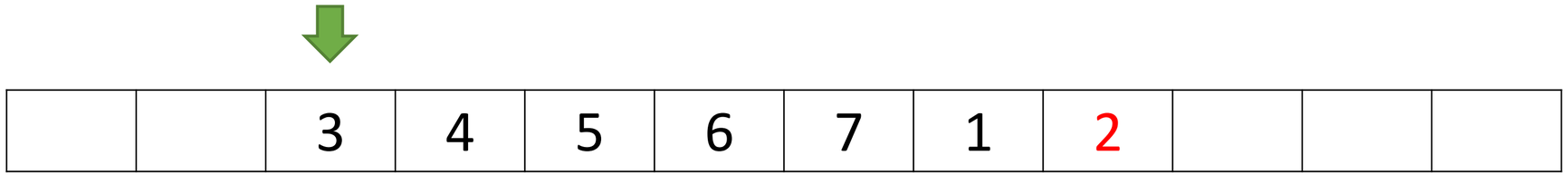


2022 Winter Algorithm Camp

1158. 요세푸스 문제

1. K번째 사람 찾기 및 제거를 큐를 통해 구현해보자.

(N, K) = (7, 3)

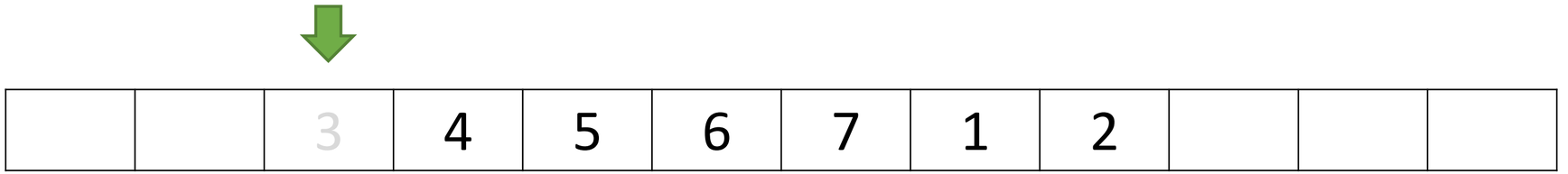


2022 Winter Algorithm Camp

1158. 요세푸스 문제

1. K번째 사람 찾기 및 제거를 큐를 통해 구현해보자.

(N, K) = (7, 3)

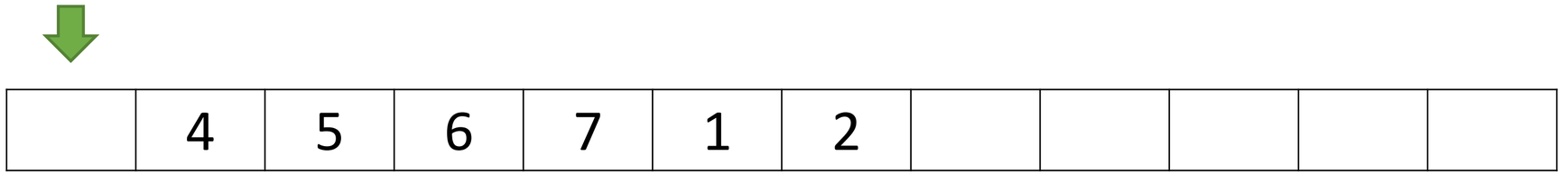


2022 Winter Algorithm Camp

1158. 요세푸스 문제

1. K번째 사람 찾기 및 제거를 큐를 통해 구현해보자.

(N, K) = (7, 3)

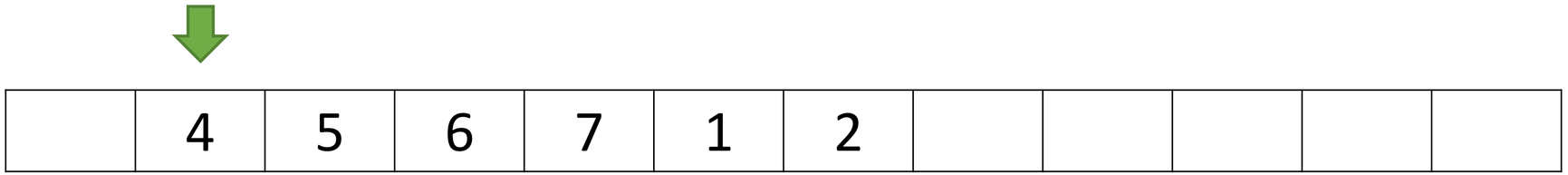


2022 Winter Algorithm Camp

1158. 요세푸스 문제

1. K번째 사람 찾기 및 제거를 큐를 통해 구현해보자.

(N, K) = (7, 3)

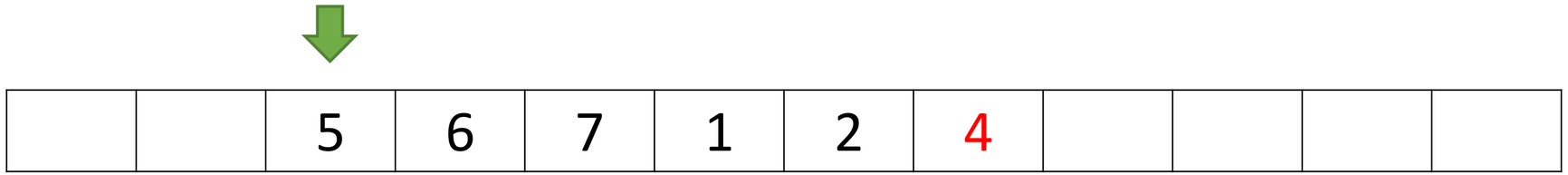


2022 Winter Algorithm Camp

1158. 요세푸스 문제

1. K번째 사람 찾기 및 제거를 큐를 통해 구현해보자.

(N, K) = (7, 3)

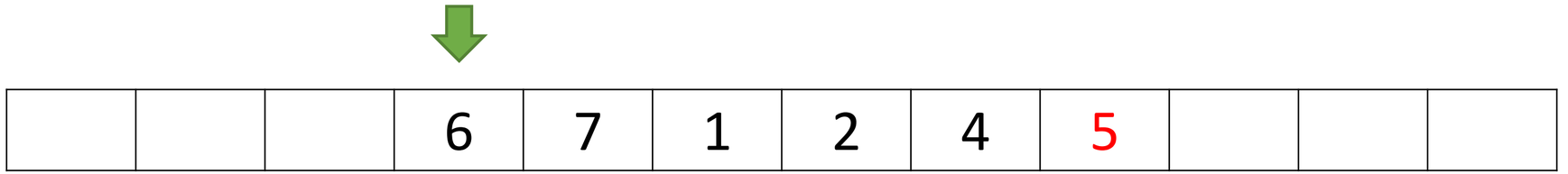


2022 Winter Algorithm Camp

1158. 요세푸스 문제

1. K번째 사람 찾기 및 제거를 큐를 통해 구현해보자.

(N, K) = (7, 3)



1158. 요세푸스 문제

2. 시간복잡도 분석

- K번째 사람을 찾고 front에서 제거하여 back에 넣을 때: O(K)
- 위 행위의 반복 횟수: N번

Thus $O(NK)$

2022 Winter Algorithm Camp

Deque – Introduction

* 소개

- 스택 + 큐
- 양쪽 끝에서 삽입과 삭제가 모두 가능한 자료구조

* Basic operations (simulation)

- push front & push back
- pop front & pop back
- front & back
- isEmpty

2022 Winter Algorithm Camp

18115. 카드 놓기

$N(1 \le N \le 10^6)$장의 카드, 1~N의 값이 중복 없이 부여

1) 제일 위의 카드 1장을 바닥에 놓기
2) 위에서 두번째 카드를 바닥에 놓기 (2장 이상일 경우 사용 가능)
3) 제일 밑에 있는 카드 1장을 바닥에 놓기 (2장 이상일 경우 사용 가능)

기술을 N번 사용하여 카드를 내려놓았을 때 결과가 $1, 2, \ldots, N$이라면 처음 상태는 무엇일까?

2022 Winter Algorithm Camp

18115. 카드 놓기

위에서도 버리고 아래에서도 제거할 수 있는 자료구조?

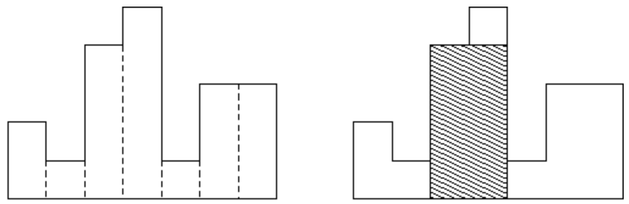
- deque이다.
- 위에서 두번째 것은 어떻게 제거할까?
  => 둘다 빼고 처음 뺀 것을 다시 위에 넣자
- 기술들을 뒤에서부터 보면서 역연산을 수행한다.

Appendix – Additional Topics

* 히스토그램에서 가장 큰 직사각형

히스토그램은 같은 너비의 직사각형 여러 개가 아래쪽으로 정렬되어 있는 도형입니다.
너비가 1인 직사각형을 붙여서 만든 도형에서 넓이가 최대인 직사각형을 찾는 문제는
매우 유명한 문제입니다.

이 문제는 여러 풀이가 존재하지만 monotone stack을 이용한 풀이는 $O(N)$만큼의
시간이 걸립니다.

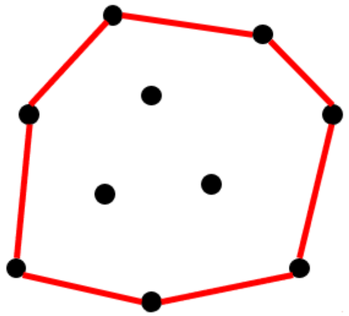


Appendix – Additional Topics

* Convex Hull

볼록 껍질(Convex Hull)은 2차원 평면 상의 여러 점들 중 일부를 이용하여 모든 점을 포함하게끔 하는 가장 작은 볼록 다각형입니다.

볼록 껍질을 구하는 알고리즘 중 그라함 스캔 알고리즘(Graham's Scan Algorithm)은 각도를 기준으로 점들을 정렬한 후 스택 구조를 이용하여 볼록 다각형의 꼭짓점에 포함시킬지 제거할지를 구합니다.



Appendix – Additional Topics

* Monotone queue

Monotone Stack과 비슷하게, queue 또한 내부적으로 monotonic한 형태를 유지함으로써 특정 문제의 해결 과정에서의 시간복잡도를 줄여주는 방식으로 응용할 수 있습니다.

이 문제의 경우, deque 내부를 monotone increasing하게 관리한다면 front에서 최솟값을 $O(1)$에 access할 수 있습니다. 구간을 한칸씩 이동하며 deque 내부에 있는 원소들의 front가 현재 보고 있는 구간에서 벗어났다면 pop_front를 해주며 특정 구간에서의 최솟값을 관리할 수 있습니다.

2022 Winter Algorithm Camp

Appendix – Problems

필수문제

6604	Matrix Chain Multiplication
1863	스카이라인 쉬운거
14713	앵무새
18115	카드 놓기
2812	크게 만들기

연습문제

9012	괄호
2841	외계인의 기타 연주
17298	오큰수
1158	요세푸스 문제
20301	반전 요세푸스
3078	좋은 친구
1935	후위 표기식2
19591	독특한 계산기
11003	최솟값 찾기