Welcome To Push_Swap!

2021.08.21
Yongjule

About push_swap

- 스택 a, b를 만들고, 정해진 operation을 사용하여 a 스택에 데이터를 넣어야함.
- Operation list
  - pa, pb : Push to stack a/b. 스택 가장 위 데이터를 반대 스택으로 넘겨줌
  - sa, sb : Swap in stack a/b. 스택 가장 위 두 데이터를 swap
  - ra, rb : Rotate in stack a/b. 스택을 한 칸씩 위로 올림.
  - rra, rrb : Reverse Rotate in stack a/b. 스택을 한 칸씩 밑으로 내림.

Flow

1. 인자 유효성 체크

2. Operation을 수행하기 위한 스택 구현

3. 정렬 - 버킷정렬, 기수정렬

4. 명령어 최적화

Check Validity

Check Validity
인자 유효성 체크

- 숫자가 아닌 인자
- INT_MIN ~ INT_MAX 범위를 벗어나는 인자
- 중복된 숫자가 들어가는 경우
- 이미 정렬된 인자

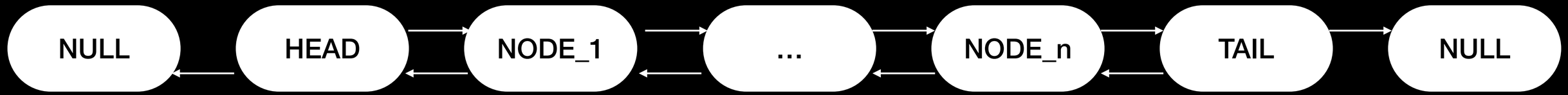
Create Stack

Create Stack

스택 구현

- push_swap 과제에서 stack이라 주어졌지만, 스택의 가장 앞/뒤 에서 데이터를 다뤄야 하므로 Deque에 가까움
- 따라서, 앞/뒤 데이터에 접근 할 수 있는 Doubly Linked List로 구현
- 이때, 각 노드에 Index를 저장한 뒤
  - INT_MIN ~ INT_MAX 범위의 숫자를 쉽게 다룰 수 있음!
  - 배열에서 Indexing은 너무 느림 -> 이진 탐색 트리 활용!

Doubly Linked List
이중 연결 리스트



NODE

```
int num;
int idx;
t_deq *next;
t_deq *prev;
```

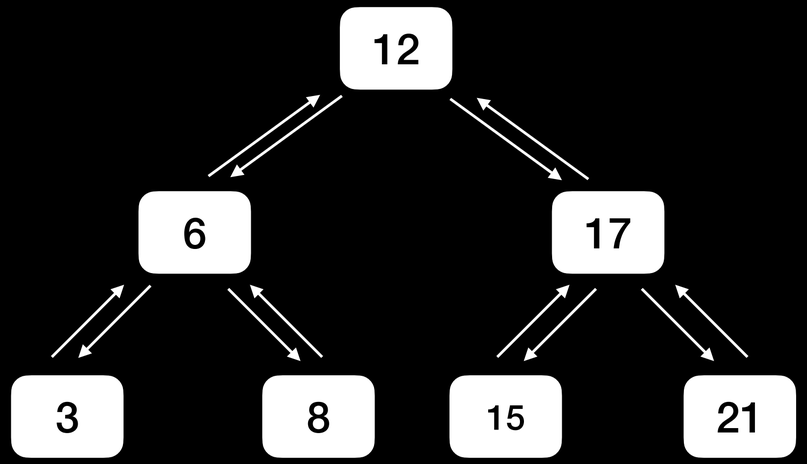
Binary Search Tree

Array vs BST(Binary Search Tree)

배열과 이진 탐색 트리의 비교

- 배열에서 indexing을 하려면, 매번 배열 내 모든 데이터와 비교해야함!
- $n + (n-1) + (n-2) + \ldots + 1 = \frac{n(n-1)}{2} \approx O(n^2)$
- 이진 탐색 트리에서는, 트리 구조를 만들면 $O(log_2 n * n)$의 시간이 걸림

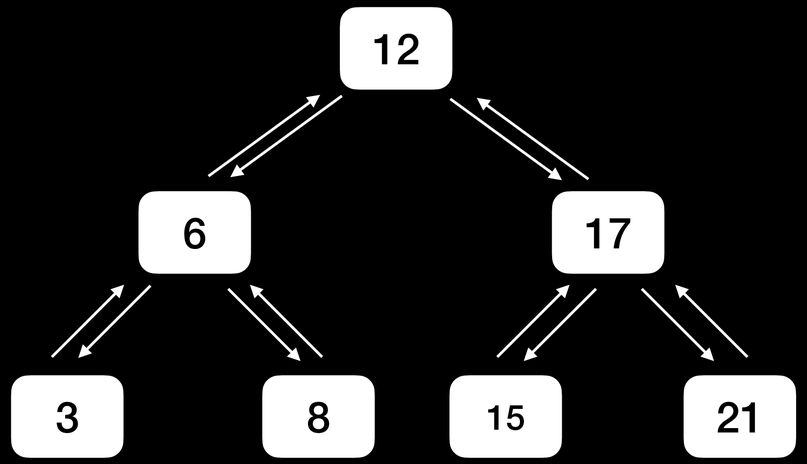
Binary Search Tree
이진 탐색 트리



- 현재 노드보다 더 작으면 왼쪽, 크면 오른쪽으로 이동
- 다음 노드가 NULL이면 그 자리에 삽입
- $h(height) = log_2 n$
- 따라서 트리를 만드는데 걸리는 시간

$$= \sum_{h=0}^{n} h = O(h^2) = O((log_2 n)^2)$$

Binary Search Tree
이진 탐색 트리



- 중위 순회를 통하여, 가장 작은 수부터 탐색 가능

$$= O(log_2 n * n)(= O(h * n))$$

- 후위 순회를 통하여 free(bst)
- 따라서 총 시간복잡도는 $O(log_2 n * n)$
- 하지만, 경우에 따라 $h = n$ 이 되는 한계가 있음.
- push_swap 에서는 Random Number를 가정하므로 해당 문제가 발생할 가능성이 낮음

Bucket Sort

Bucket Sort
버킷 정렬

- 데이터를 여러 Bucket에 담고, 각 Bucket에서 정렬을 함.
- 이때, 각 Bucket에서 수행하는 정렬 알고리즘은 주로 Insertion Sort
- 각 스택을 Bucket으로 보고, Bucket Sort를 하면 명령어 최적화가 용이할 것이라 기대
- 한계 - 제한된 Bucket의 수, Bucket 안에 너무 많은 데이터가 있음, Insertion Sort가 Push_swap에서 상당히 많은 Operation을 필요로 함

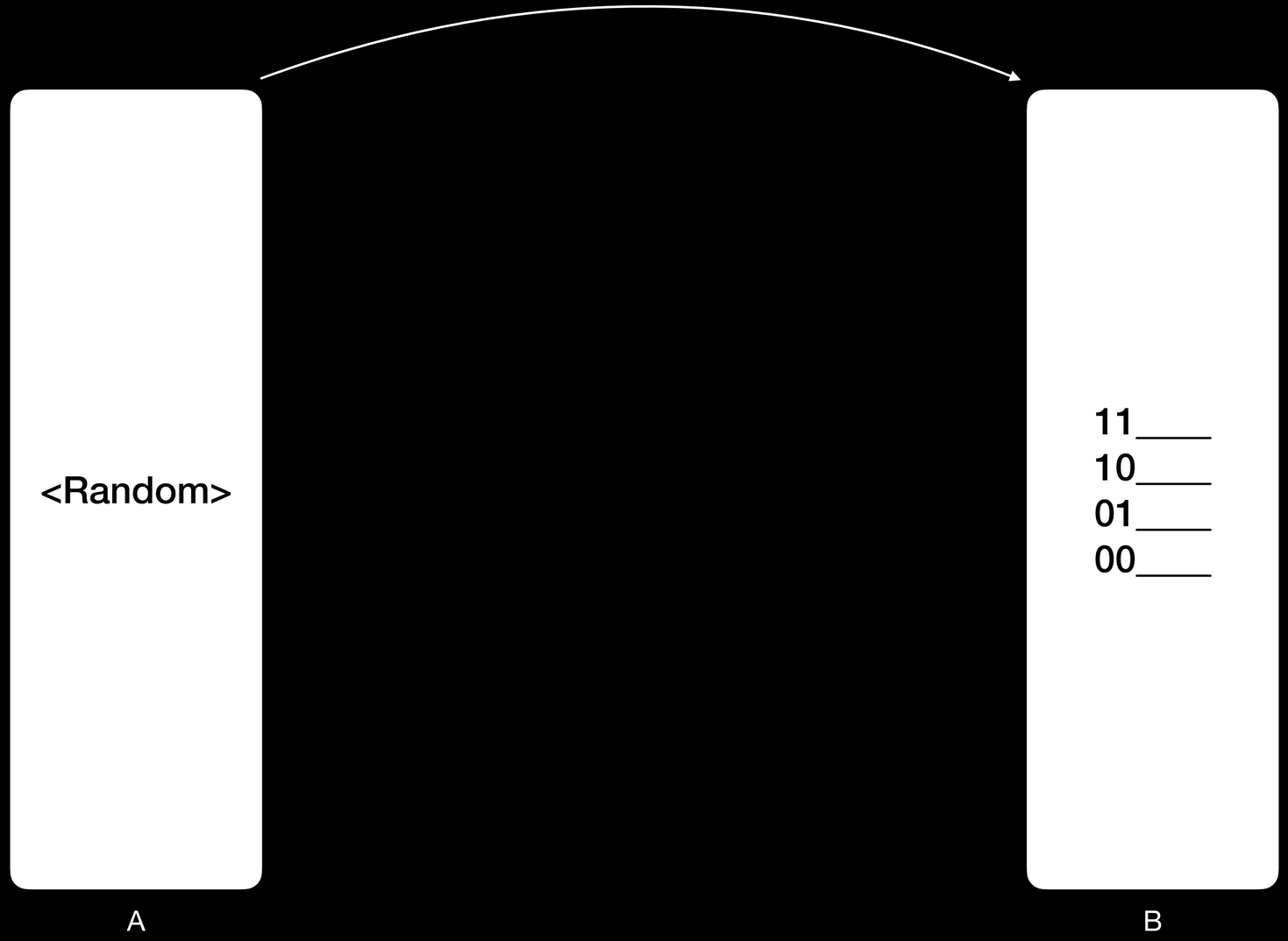
Radix Sort

Radix Sort
기수 정렬

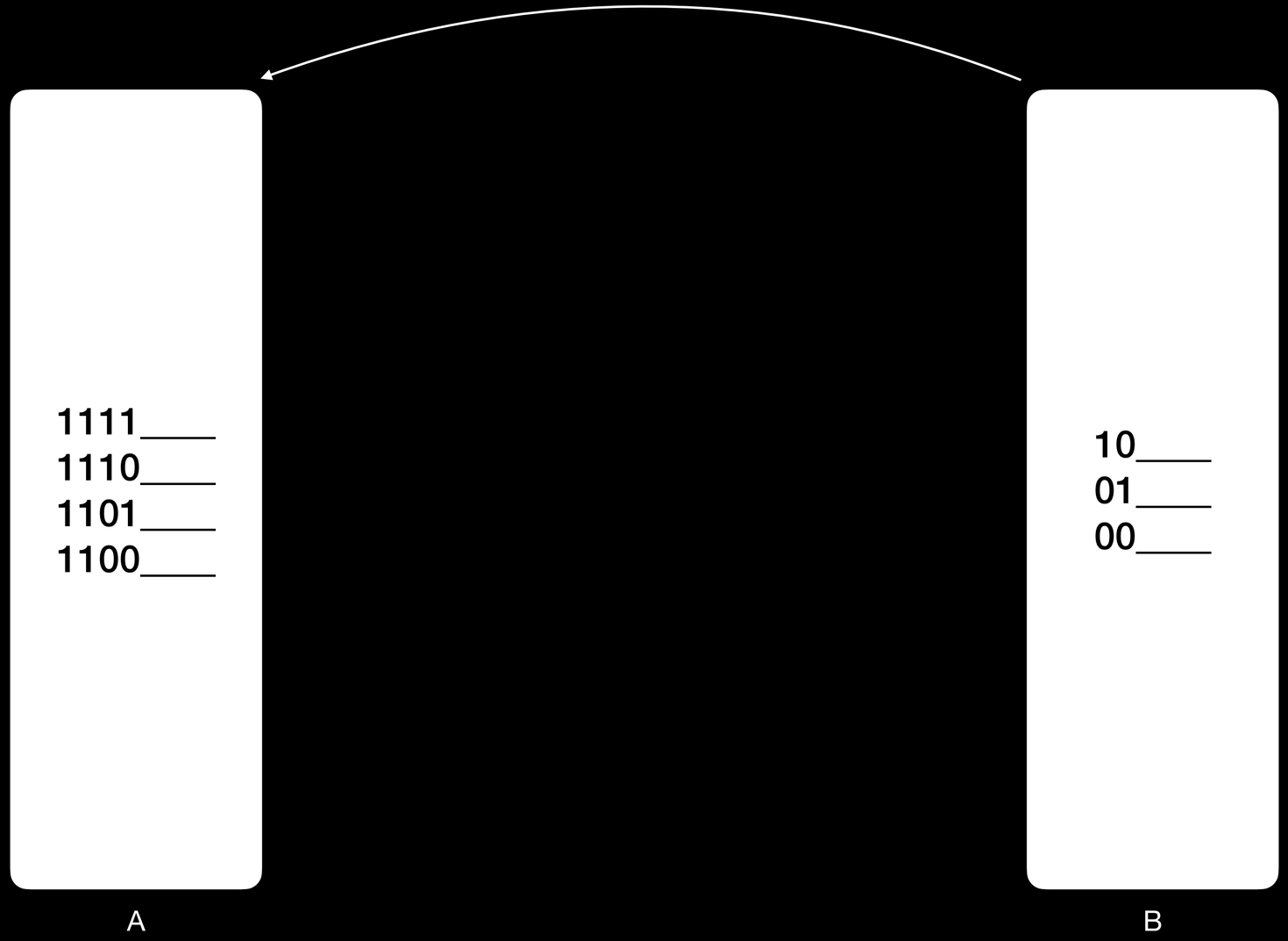
- 자릿수 기반으로 정렬하는 알고리즘
- MSD(Most Significant Digit) / LSD(Least Significant Digit)로 나뉨
- $O((n + b) * log_b(k))$

Radix Sort in push_swap

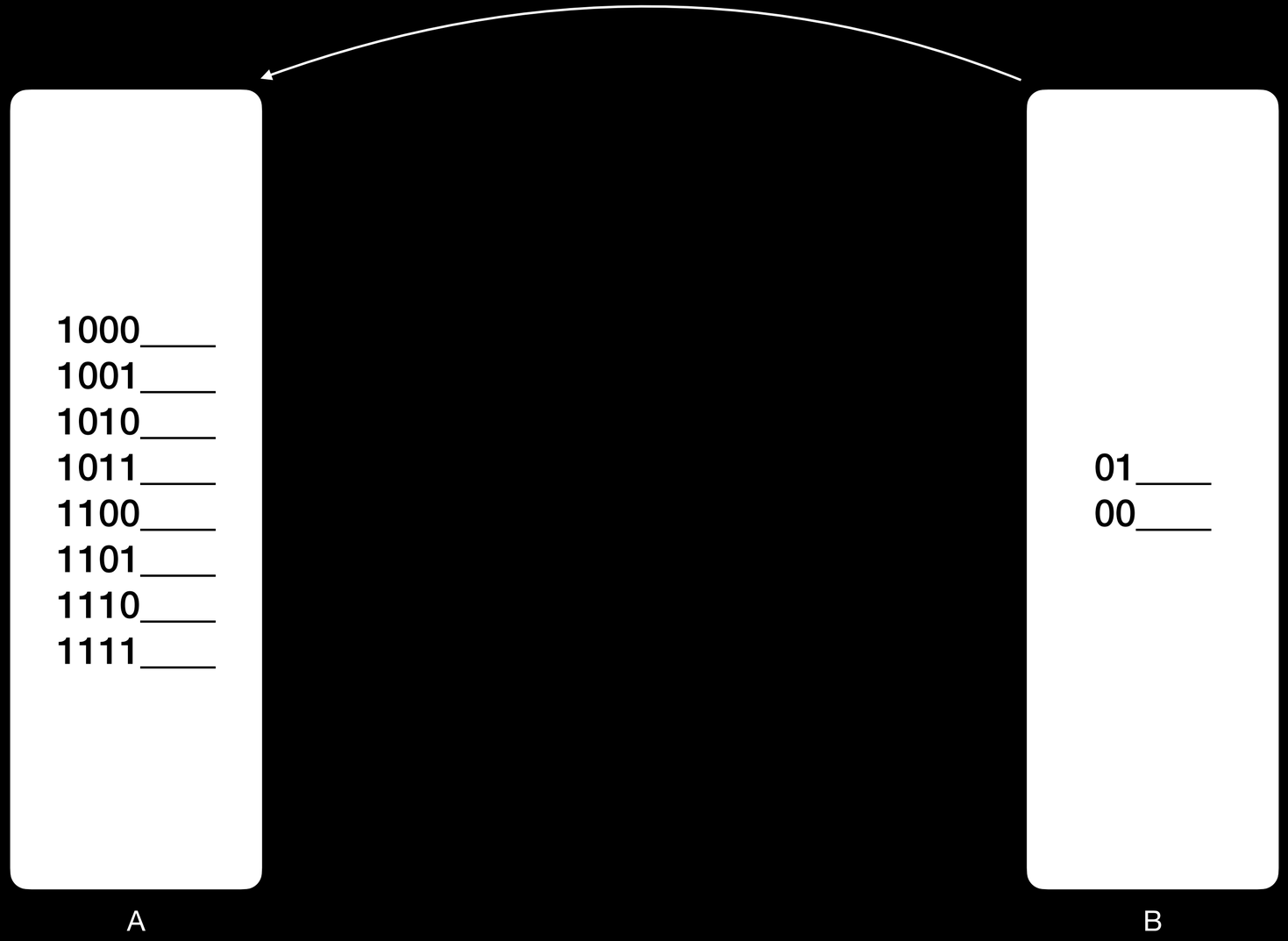
Radix Sort in push_swap



Radix Sort in push_swap



Radix Sort in push_swap



Radix Sort in push_swap