땅울림

07

우선순위 큐
Priority Queue

자료구조 스터디 7회차

- 우선순위 큐
- 삽입 정렬 & 선택 정렬
- 문제 유형들

1

우선순위 큐 개념

데이터를 정렬된 순서로(우선순위대로) 저장

가장 높은 우선순위를 가진 데이터가 항상 맨 앞에 위치
가장 높은 우선순위를 가진 데이터에 빠르게 접근할 수 있음

데이터를 추가할 때는 우선순위에 맞게 위치에 추가되며
삭제할 때는 가장 높은 우선순위를 가진 데이터가 먼저 삭제된다.

자료 구조	삭제되는 요소
스택	가장 최근에 들어온 데이터
큐	가장 먼저 들어온 데이터
우선순위 큐	가장 우선순위가 높은 데이터

2

우선순위 큐를 이용한 정렬

문제 유형

삽입 정렬

수열에서 PQ로 **삽입**할 때 정렬함

	수열 S	우선순위 큐 P
입력	(7, 4, 8, 2, 5, 3, 9)	( )
단계1 (a)	(4, 8, 2, 5, 3, 9)	(7)
(b)	(8, 2, 5, 3, 9)	(4, 7)
(c)	(2, 5, 3, 9)	(4, 7, 8)
(d)	(5, 3, 9)	(2, 4, 7, 8)
(e)	(3, 9)	(2, 4, 5, 7, 8)
(f)	(9)	(2, 3, 4, 5, 7, 8)
(g)	( )	(2, 3, 4, 5, 7, 8, 9)
단계2 (a)	(9)	(2, 3, 4, 5, 7, 8)
(b)	(9, 8)	(2, 3, 4, 5, 7)
(c)	(9, 8, 7)	(2, 3, 4, 5)
(d)	(9, 8, 7, 5)	(2, 3, 4)
(e)	(9, 8, 7, 5, 4)	(2, 3)
(f)	(9, 8, 7, 5, 4, 3)	(2)
(g)	(9, 8, 7, 5, 4, 3, 2)	( )

PQ에 넣을 때 정렬

<정렬된 PQ>

정렬된거 그대로 다시 수열에 넣기

선택 정렬

PQ에 정렬 안하고
그냥 다 넣어버리고
가장 높은 우선순위의
값을 하나씩 **선택**해서
수열에 넣으면서 정렬

	수열 S	우선순위 큐 P
입력	(7, 4, 8, 2, 5, 3, 9)	( )
단계1 (a)	(4, 8, 2, 5, 3, 9)	(7)
(b)	(8, 2, 5, 3, 9)	(7, 4)
(c)	(2, 5, 3, 9)	(7, 4, 8)
(d)	(5, 3, 9)	(7, 4, 8, 2)
(e)	(3, 9)	(7, 4, 8, 2, 5)
(f)	(9)	(7, 4, 8, 2, 5, 3)
(g)	( )	(7, 4, 8, 2, 5, 3, 9)
단계2 (a)	(9)	(7, 4, 8, 2, 5, 3)
(b)	(9, 8)	(7, 4, 2, 5, 3)
(c)	(9, 8, 7)	(4, 2, 5, 3)
(d)	(9, 8, 7, 5)	(4, 2, 3)
(e)	(9, 8, 7, 5, 4)	(2, 3)
(f)	(9, 8, 7, 5, 4, 3)	(2)
(g)	(9, 8, 7, 5, 4, 3, 2)	( )

정렬없이 PQ에 넣기

<정렬되지 않은 PQ>

다시 수열에 넣을 때 정렬

〈 main() 함수 〉

'수열을 저장할 벡터'와 '우선순위큐'를 활용해서 정렬

```
for (0부터 n까지 반복) {
    우선순위큐.push(수열벡터.front());
    수열벡터.erase(v.begin());
}
```

1 정렬 수열 벡터가 빌 때까지 요소를 우선순위 큐에 삽입

```
for (0부터 n까지 반복) {
    수열벡터.push_back(우선순위큐.pop());
}
```

2 옮김 우선순위 큐가 빌 때까지 요소를 다시 수열벡터에 삽입

2-1

우선순위 큐를 이용한 삽입 정렬 구현

문제 유형

삽입 정렬

리스트에 **삽입**할 때 미리 정렬시키면서 삽입하고
맨 앞에서부터 꺼내면서 다시 옮기는 방식

Sorted List (**정렬된** 벡터)로 구현

삽입
O(n) : 자기가 들어갈 위치(정렬된 위치)를 찾아야 함
Insert

삭제
O(1) : 맨 앞의 수(가장 작은 수)를 삭제하면 끝
removeMin

삽입 정렬 : 우선순위 큐를 sorted list로 구현했을 때 활용되는 알고리즘

〈PQ의 push연산〉

```
if (비어있으면) {
    수열벡터.push_back(data)
} else {
    vector<int>::iterator iter
    for(iter반복자 수열벡터 순회) {
        if(우선순위 조건) {
            수열벡터.insert(iter, data)
            return
        }
    }
    수열벡터.insert(iter, data)
}
```

1 정렬 시퀀스가 빌 때까지 요소를 우선순위 큐에 삽입

- insert할 때 정렬된 자신의 자리를 찾는 과정 n번

2 옮김 우선순위 큐가 빌 때까지 요소를 다시 시퀀스에 삽입

- 1단계에서 이미 정렬 완료된 것을 그대로 다시 시퀀스에 넣어주기 n번

중위 순회

삽입 정렬 : 우선순위 큐를 sorted list로 구현했을 때 활용되는 알고리즘

1 정렬 시퀀스가 빌 때까지
요소를 우선순위 큐에 삽입

- insert할 때 정렬된 자신의 자리를 찾는 과정 n번

〈PQ의 pop연산〉

```
if (비어있지 않으면) {
   data = 수열벡터.front();
   수열벡터.erase(수열벡터.begin())
   return data
}
```

2 옮김 우선순위 큐가 빌 때까지
요소를 다시 시퀀스에 삽입

- 1단계에서 이미 정렬 완료된 것을 그대로 다시 시퀀스에 넣어주기 n번

2-2

우선순위 큐를 이용한 선택 정렬 구현

문제 유형

선택 정렬

리스트에 정렬 따위 없이 다 그냥 넣어버리고
최소값을 하나씩 **선택**해서 꺼내면서 정렬하는 방식

Unsorted List (**정렬되지 않은** 벡터)로 구현

삽입 O(1) : 그냥 맨 앞에 넣음(just 옮김)
Insert

삭제 O(n) : 가장 작은 수를 찾아야 함
removeMin

선택 정렬 : 우선순위 큐를 unsorted list로 구현했을 때 활용되는 알고리즘

〈PQ의 push연산〉

수열벡터.push_back(data);

1 옮김 시퀀스가 빌 때까지 요소를 우선순위 큐에 삽입

- insert 연산 활용 PQ에 그대로 카피 (정렬안하고 그대로 삽입) n번

2 정렬 우선순위 큐가 빌 때까지 요소를 다시 시퀀스에 삽입

- 정렬안된 배열에서 최솟값을 찾아내고 맨 앞에 배치하는 과정 n번

선택 정렬 : 우선순위 큐를 unsorted list로 구현했을 때 활용되는 알고리즘

〈PQ의 pop연산〉

```
vector〈int〉::iterator p
for(반복자 iter선언 후 수열벡터 순회) {
    if(우선순위 조건) {
        p반복자 업데이트
    }
  int priorityValue = *p
  수열벡터.erase(p)
  return priorityValue;
}
```

1 옮김 시퀀스가 빌 때까지 요소를 우선순위 큐에 삽입

- insert 연산 활용 PQ에 그대로 카피 (정렬안하고 그대로 삽입) n번

2 정렬 우선순위 큐가 빌 때까지 요소를 다시 시퀀스에 삽입

- 정렬안된 배열에서 최솟값을 찾아내고 맨 앞에 배치하는 과정 n번

선택 정렬

Unsorted List
비정렬 벡터로 구현

삽입
O(1) fast

삭제
O(n)

삽입 정렬

Sorted List
정렬된 벡터로 구현

삽입
O(n)

삭제
O(1) fast

3

우선순위 큐 문제 유형

문제 유형

21년도 실습 문제 족보

https://github.com/Landvibe-DataStructure-2023Study/LimJumin/tree/main/21%20%EC%8B%A4%EC%8A%B5%20%EC%BD%94%EB%93%9C/%EC%9A%B0%EC%84%A0%EC%88%9C%EC%9C%84%20%ED%81%90

P1	P2	P3	P4
수열 입력 받고, PQ 이용한 삽입 정렬 구현 (오름차순 정렬)	수열 입력 받고, PQ 이용한 선택 정렬 구현 (내림차순 정렬)	수열 입력 받고, PQ 이용한 선택 정렬 구현 (오름차순 정렬)	수열 입력 받고, PQ 이용한 삽입 정렬 구현 (내림차순 정렬)

22년도 실습 문제 족보

https://github.com/Landvibe-DataStructure-2023Study/LimJumin/tree/main/22%20%EC%8B%A4%EC%8A%B5%20%EC%BD%94%EB%93%9C/%EC%9A%B0%EC%84%A0%EC%88%9C%EC%9C%84%20%ED%81%90

P1	P2	P3	P4
정렬된 벡터로 구현된 PQ 이용, 삽입 정렬로 오름차순으로 정렬	비정렬 벡터로 구현된 PQ 이용, 조건에 맞춰 우선순위 결정하고 우선순위 높은 순으로 출력	비정렬 벡터로 구현된 PQ 이용, 선택 정렬로 내림차순으로 정렬	비정렬 벡터로 구현된 PQ 이용, 조건에 맞춰 우선순위 결정하고 우선순위 높은 순으로 출력

트리 순회 문제 유형