2주차 우선순위 큐

통계 수정 삭제

iming03 · 방금 전 · 비공개

0

알고리즘 우선순위 큐 자료구조

알고리즘 스터디

목록 보기 2/2

문제: 백준 2696번 중앙값 구하기

어떤 수열을 읽고, 홀수번째 수를 읽을 때 마다, 지금까지 입력받은 값의 중앙값을 출력하는 프로그램을 작성하시오.
예를 들어, 수열이 1, 5, 4, 3, 2 이면, 홀수번째 수는 1번째 수, 3번째 수, 5번째 수이고, 1번째 수를 읽었을 때 중앙값은 1, 3번째 수를 읽었을 때는 4, 5번째 수를 읽었을 때는 3이다.

입력

첫째 줄에 테스트 케이스의 개수 T(1 ≤ T ≤ 1,000)가 주어진다. 각 테스트 케이스의 첫째 줄에는 수열의 크기 M(1 ≤ M ≤ 9999, M은 홀수)이 주어지고, 그 다음 줄부터 이 수열의 원소가 차례대로 주어진다. 원소는 한 줄에 10개씩 나누어져있고, 32비트 부호있는 정수이다.

출력

각 테스트 케이스에 대해 첫째 줄에 출력하는 중앙값의 개수를 출력하고, 둘째 줄에는 홀수 번째 수를 읽을 때 마다 구한 중앙값을 차례대로 공백으로 구분하여 출력한다. 이때, 한 줄에 10개씩 출력해야 한다.

예제 입력 1 복사

```
3
9
1 2 3 4 5 6 7 8 9
9
9 8 7 6 5 4 3 2 1
23
23 41 13 22 -3 24 -31 -11 -8 -7
3 5 103 211 -311 -45 -67 -73 -81 -99
-33 24 56
```

예제 출력 1 복사

```
5
1 2 3 4 5
5
9 8 7 6 5
12
23 23 22 22 13 3 5 5 3 -3
-7 -3
```

우선순위 큐란?

우선순위 큐는 여러가지 데이터를 저장하는 데이터 구조 중 하나이다.

우선순위 큐 ADT(추상자료형)은 항목들을 저장하는데 각 항목은 (키, 원소)쌍으로 되어있습니다. 응용에는 탑승 대기자, 옥션(경매), 주식시장 등이 있습니다.

우선순위 큐 ADT 메소드

- 주요 메소드
  - insertitem(k, e): 키 k인 원소 e를 큐에 삽입
  - element removeMin(): 최소키 원소 삭제, 반환
- 일반 메소드
  - integersize(): 큐 항목 수 반환
  - boolean isEmpty(): 큐가 비어있는지 여부 반환
- 접근 메소드
  - element minElement(): 큐에서 최소키를 가진 원소 반환
  - element minKey(): 큐에서 최소키 반환
- 예외
  - emptyQueueException(): 비어있는 큐에 대해 삭제/원소 접근 시도할 경우 발령
  - full QueueException(): 만원 큐에 대해 삽입을 시도할 경우 발령

우선순위 큐를 이용한 정렬

- 비교 가능한 원소 집합 정렬에 우선순우 큐 이용가능

1. 연속적인 **insertItem(e, e)** 작업을 통해 원소들 **하나씩 삽입**(key=e로 전제)
2. 연속적인 **removeMin()** 작업을 통해 원소들을 **정렬 순서로 삭제**

- 실행시간: 우선순위 큐의 구현에 따라 다른

리스트에 기초한 우선순위 큐

1. 무순리스트(순서X)
    - 임의 순서로 저장
    - 성능
        - **insertItem - O(1)** 시간(상수 시간): 맨뒤/맨앞에 삽입가능
        - **removeMin, minKey, minElement - O(n)** 시간(선형시간): 최소 키를 찾기 위해 전체 리스트를 순회해야함
    - =**선택정렬**(select-sort)
    - PQ-sort의 일종
    - 실행 시간: n회의 insertItem 작업 후 삽입하는데 O(n)시간 소요, n회의 removeMin으로 삭제하는데 n+(n-1)+(n-2)+...+2+1에 비례하는 시간 소요 -> **Total: O(n^2)**
2. 순서리스트(순서O)
    - 키 정렬 순서로 저장
    - 성능
        - **insertItem - O(n)** 시간(선형 시간): 삽입할 곳을 탐색해야하므로
        - **removeMin, minKey, minElement - O(n)** 시간(상수시간): 최소키가 리스트 맨 앞에 있음
    - =삽입정렬(insertion-sort)
    - PQ-sort의 일종
    - 실행 시간: n회의 insertItem 작업 후 삽입하는데 n+(n-1)+(n-2)+...+2+1에 비례하는 시간 소요, n회의 removeMin으로 삭제하는데 O(n)시간 소요-> Total: O(n^2)

"제자리"에서 할 수 있나?(제 2공간 사용X): 제자리 선택 정렬 & 제자리 삽입 정렬

- selection-sort, insertion-sort 모두 O(n) 공간 외부의 우선순위 큐 사용해 리스트 정렬
- O(1) 공간만 사용 -> 제자리에서 수행 : 오직 상수 메모리만 사용

선택정렬 vs 삽입정렬

공통점: 전체적으로 O(n^2)시간

- 내부 반복문: O(n) 선형 탐색
- 외부 반복문: O(n) 패스
- 제자리 버전은 O(1) 공간 소요
- 작은 n에 대해 유용

1. 초기 리스트가 완전히/거의 정렬된 경우
    - in-place **insertion**-sort(삽입정렬)가 더 빠름
    - 내부 반복문이 O(1)시간 소요 -> 따라서 전체적으로 O(n)시간 수행
2. **swapElements** 작업이 비싼 경우
    - in-place **selection**-sort(선택정렬)가 더 빠름
    - **swap Elements** 작업이 패스마다 O(1)시간 수행되는데 반해, in-place insertion-sort에서는 동일 작업이 패스마다 최악의 경우 O(n) 시간 수행되므로

성능 요약

우선순위 큐		작업 수행시간		정렬방식
	insertItem	removeMin	minKey, minElement	
무순리스트	O(1)	O(n)	O(n)	선택정렬
순서리스트	O(n)	O(1)	O(1)	삽입정렬

문제 풀이

아이디어: 최대 힙(MaxHeap)의 최댓값과 최소 힙(MinHeap)의 최솟값의 중간값 이용

여기에서 우선순위 큐와 최대 힙의 개념이 헷갈려서 챗 gpt에게 물어봤다.

최대 힙(Max Heap)과 우선순위 큐(Priority Queue)는 개념적으로 유사하지만 구현 방식과 사용 목적에 약간의 차이가 있습니다.

<구현 방식>:

최대 힙 (Max Heap): 이진 트리 구조로, 부모 노드가 항상 자식 노드보다 크거나 같은 값을 가지는 자료 구조입니다. 가장 큰 값이 루트 노드에 위치하며, 추가되는 값은 항상 부모 노드보다 작으면서 힙 속성을 유지합니다.

우선순위 큐 (Priority Queue): 다양한 방식으로 구현될 수 있으며, 우선순위에 따라 요소를 관리하는 자료 구조입니다. 최대 힙, 최소 힙 또는 이진 힙 등을 기반으로 우선순위 큐를 구현할 수 있습니다. 최대 힙은 우선순위 큐의 하나의 구현 방식일 뿐입니다.

<사용 목적>

최대 힙 (Max Heap): 주로 최대값을 빠르게 찾아야 하는 경우에 사용됩니다. 예를 들어, 작업 스케줄링, 그래프 알고리즘 (최단 경로 찾기), 힙 정렬 등에서 활용됩니다.
우선순위 큐 (Priority Queue): 다양한 우선순위에 따라 데이터를 처리해야 할 때 사용됩니다. 최소값, 최대값, 높은 우선순위 또는 낮은 우선순위의 요소를 빠르게 처리하고 싶을 때 유용합니다. 예를 들어, 태스크 스케줄링, 데이터 압축, 다익스트라 알고리즘 등에서 사용됩니다.

최대 힙은 우선순위 큐의 한 형태로 볼 수 있으며, 우선순위 큐는 다양한 문제에 적용할 수 있는 추상 데이터 타입입니다. 우선순위 큐의 구현은 최대 힙 외에도 최소 힙, Fibonacci 힙, 이진 힙, 이진 탐색 트리 등 다양한 자료 구조를 활용하여 이루어질 수 있습니다. 선택한 구현 방식은 주어진 문제의 특성과 요구 사항에 따라 다를 것입니다.

최대 힙이 우선순위 큐의 한 형태라고 생각하고 다시 보니 문제가 더 쉬워보였다.
최대힙을 구현하고 그 중간 값을 구해야겠다고 생각했다.

문제 코드

노드를 사용하는 코드가 가장 약해서 많이 막혔고 계속 chatgpt에게 물어봤다...
다음에는 온전한 내힘으로 풀도록 열심히 노력해야겠다...

강민돌

민돌이의 공부

이전 포스트
1주차 자료구조 복습

0개의 댓글

댓글을 작성하세요

댓글 작성