자료구조

숙제7 과제

우선순위큐

2018187116

컴퓨터공학부

김용현

〈MinHeap〉

```cpp
#include <iostream>
#define ELEM 200
using namespace std;

class heapnode {
private:
    int key;

public:
    /* Constructor */
    heapnode(int _key = 0) : key(_key) {}

    /* Method */
    void setkey(int _key) {
        key = _key;
    }
    int getkey(void) {
        return key;
    }
    void display(void) {
        cout << key << ' ';
    }
};

class minheap {
private:
    heapnode node[ELEM];
    int size;

public:
    /* Constructor */
    minheap() : size(0) {}

    /* Method */
    heapnode& getparent(int i) {
        return node[i / 2];
    }
    heapnode& getleft(int i) {
        return node[i * 2];
    }
    heapnode& getright(int i) {
        return node[i * 2 + 1];
    }


    bool empty() {
        return size == 0;
    }
    bool full() {
        return size == ELEM - 1;
    }
    void push(int key) { //insert
        if (full()) return;

        int i = ++size;

        while (i != 1 && key < getparent(i).getkey()) {
            node[i] = getparent(i);
            i /= 2;
        }
        node[i].setkey(key);
    }
    heapnode pop() { //remove
        heapnode item = node[1];
        heapnode last = node[size--];
        int parent = 1;
        int child = 2;

        while (child <= size) {
            if (child < size && getleft(parent).getkey() > getright(parent).getkey())
                child++;
            if (last.getkey() <= node[child].getkey())
                break;

            node[parent] = node[child];
            parent = child;
            child *= 2;
        }

        node[parent] = last;
        return item;
    }
    heapnode top() {
        return node[1];
    }
    void display() {
        for (int i = 1; i <= size; i++)
            node[i].display();
        cout << '\n';
    }
};

int main(int argc, char* argv[]) {
    /* Faster */
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    /* Init */
    minheap mh;

    /* Input */
    int n, k; cin >> n >> k;
    for (int i = 0; i < n; i++) {
        int x; cin >> x;
        mh.push(x);
    }

    /* Output1 */
    mh.display();

    /* Pop */
    for (int i = 0; i < k; i++)
        mh.pop();

    /* Output2 */
    mh.display();

    /* Return */
    return 0;
}
```

〈헤더파일〉

#. 기본적인 iostream만 포함시켜 주었다.

〈class heapnode〉

#. 교과서에 있는 코드 그대로에서, 가독성만 좋게 바꾸어 작성하였다.

〈class minheap〉

#. 교과서에 있는 maxheap코드에서, upheap · downheap연산 시 부등호 방향만 바꾸어 minheap을 만들어 주었다.

〈Faster〉

#. 싱글 쓰레드 환경에서, printf와 scanf 함수가 이용하는 stdio.h 버퍼와 cin과 cout이 이용하는 iostream의 버퍼의 동기화를 해제함으로써, 입출력 속도를 빠르게 하였다.
ios::sync_with_stdio(0)은 stdio.h와 iostream의 동기화를 해제하겠다는 뜻이다.
cin.tie(0) 및 cout.tie(0)의 경우, cin과 cout은 서로 연결되어 있어 cin을 쓰면 출력 버퍼를 비우고 입력이 발생한다. 이러한 flush(버퍼를 비우는 과정) 과정도 시간이 들기 때문에, cin과 cout의 상호 연결을 끊어주기 위하여 해당 코드를 삽입하는 것이다.

〈Init〉

#. minheap mh를 선언하였다.

〈Input〉

#. 입력으로 주어진 값을 입력 받아 minheap에 집어넣는다.

〈Output1〉

#. minheap 내용을 출력한다.

〈Pop〉

#. minheap에서 값을 k개 만큼 제거한다.

〈Output2〉

#. minheap 내용을 출력한다.

〈Return〉

#. 0값을 반환한다.

〈힙 조건 검사〉

```cpp
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

bool DC(vector<int>& v, int i) {
    /* Base Case */
    if (i * 2 + 1 >= v.size() && i * 2 + 2 >= v.size()) //말단노드
        return true;

    /* Inductive Step */
    if (i * 2 + 2 < v.size())   //자식 노드가 2개인 경우.
        return
        v[i] >= v[i * 2 + 1] &&
        v[i] >= v[i * 2 + 2] &&
        DC(v, i * 2 + 1) &&
        DC(v, i * 2 + 2);
    else                        //자식 노드가 1개인 경우.
        return
        v[i] >= v[i * 2 + 1];
}

int main(int argc, char* argv[]) {
    /* Faster */
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    /* Input */
    int n; cin >> n;
    vector<int> v(n, 0); for (auto& i : v) cin >> i;

    /* Output */
    if (DC(v, 0))
        cout << "YES" << '\n';
    else
        cout << "NO" << '\n';

    /* Return */
    return 0;
}
```

〈헤더파일〉

#. 입출력을 위한 iostream과, 벡터를 사용하기 위한 vector를 include 해주었다.

〈DC〉

#. 재귀 방식으로 힙 조건을 검사 하는 함수. Base Case의 경우, 말단노드인지 여부를 판단해 return 시켜준다. Inductive Step의 경우, 자식노드가 2개인 경우와 자식노드가 왼쪽자식노드 1개인 경우를 나누어 재귀를 호출하였다. 왼쪽 · 오른쪽 자식노드보다 값이 큰지 아닌지 여부를 판단 후, 왼쪽 · 오른쪽 자식 노드에 대해서도 같은 일을 재귀를 통해 반복해준다.

〈Faster〉

#. 싱글 쓰레드 환경에서, printf와 scanf 함수가 이용하는 stdio.h 버퍼와 cin과 cout이 이용하는 iostream의 버퍼의 동기화를 해제함으로써, 입출력 속도를 빠르게 하였다.
ios::sync_with_stdio(0)은 stdio.h와 iostream의 동기화를 해제하겠다는 뜻이다.
cin.tie(0) 및 cout.tie(0)의 경우, cin과 cout은 서로 연결되어 있어 cin을 쓰면 출력 버퍼를 비우고 입력이 발생한다. 이러한 flush(버퍼를 비우는 과정) 과정도 시간이 들기 때문에, cin과 cout의 상호 연결을 끊어주기 위하여 해당 코드를 삽입하는 것이다.

〈Input〉

#. 입력으로 주어진 값을 n과 vector에 각각 입력받는다.

〈Output〉

#. DC가 true인 경우, YES. DC가 false인 경우, NO를 반환한다.

〈Return〉

#. 값을 반환한다.