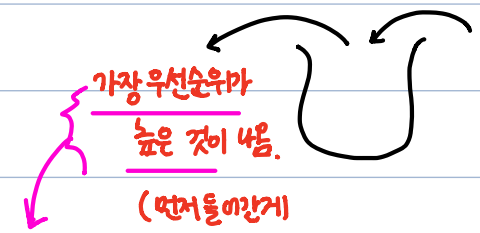
〈 우선순위 큐 〉

1) 정의

우선순위 큐 : 원소를 제거할 시, 가장 우선순위가 높은 원소를 제거



ex) 가장 큰 것 = 우선순위. ∴ 9.push(1)
9.push(2)
9.push(3)
9.push(5)

9.pop() 하면 5가 나옴.

[0] [1] [2] [3] [4] [5] [6] [7]

myQueue | 1 | 4 | 3 | 5 | | | | |

우선순위제일 높은거 찾아서 ⇒ 5 pop

그다음 〃 ⇒ 4 pop

⋮

push 연산 : 그냥 뒤에다가 하나 추가하면 되니까 시간복잡도 : O(1)

pop 연산 : 우선순위 제일 큰 애를 찾기 → 찾은 애 빼고 → 남은 애들 앞으로 당겨오기

O(n) O(n)

O(2n) ≒ O(n)

2) 구현 [예제2] 우선순위 큐 구현하기

* 배열을 이용하여 우선순위 큐를 구현하시오

시스템 입력의 예

```
priorityQueue myPQ;

myPQ.push(1)
myPQ.push(4)
myPQ.pop()

print myPQ.top()
```

출력의 예

```
4
```

→ 우선순위가 가장 높은 원소 반환

```cpp
#include <stdio.h>

//우선순위 큐 구현하기
const int MAX = 100;

struct prioirityQueue{
  //    0 1 2 3 4 5 6 7 8
  //data

  int data[MAX];
  int len = 0;

  void push(int x){
    data[len++] = x;
  }

  void pop(){
    //1. 우선순위 가장 높은 원소를 찾는다.
    int myMax = -10000000000, myMaxinx = -1;
    for(int i=0; i<len; i++){
      if(data[i]>myMax){
        myMax = data[i];//우선순위 가장 높은 것 알 수 있음
        myMaxinx = i;//위치(index)알 수 있음
      }
    }
    //2. 그 원소를 제거하고
    //(즉 제거하려면 값만 알아선 안되고 그 값의 위치(index)를 알아야 한다.
    for(int i=myMaxinx; i<len; i++){
      data[i] = data[i+1];//3. 뒤의 원고를 앞으로 당긴다.
    }
    len--;//원소 하나 제거되었으니까 전체 길이를 하나 줄여야 함

  }

  int top(){
    int myMax = -10000000000;

    for(int i=0; i<len; i++){
      if(data[i]>myMax){
        myMax = data[i];//우선순위 가장 높은 것 알 수 있음
      }
    }
    return myMax;
  }

};


int main() {

  prioirityQueue myPQ;

  myPQ.push(1);
  myPQ.push(8);
  myPQ.push(7);
  myPQ.push(5);

  printf("%d\n", myPQ.top());

  myPQ.pop();

  printf("%d\n", myPQ.top());

  return 0;
}
```

3) 우선순위 큐의 효율성

```
int main() {
  prioirityQueue myPQ;

  for(int i=100000; i>=1; i--){
    myPQ.push(i); //O(1)
  }

  for(int i=1; i<=100000; i++){
    myPQ.pop(); //O(n^2) --> 엄청 큰 시간복잡도이다.
  }

  //이 두 연산을 얼마나 빠르게 하는지 보자.
  //15-16초 정도 걸린다.

  // 그래서 , 이로부터 배열을 통해서 우선순위큐를 구현하는게
  //효율적인 구현이 아닐 수도 있다는 것을 알 수 있다.
  printf("%d\n", myPQ.data[1]);

  return 0;
}
```