컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 31강 - 우선순위 큐

Fast campus

학습 목표

우선순위 큐

1) 우선 순위 큐의 필요성과 개념에 대해서 이해합니다.
2) 우선 순위 큐를 C언어를 이용해 구현하는 방법에 대해서 학습합니다.

우선순위 큐

우선순위 큐의 필요성

우선순위 큐는 '우선 순위'를 가진 데이터들을 저장하는 큐를 의미합니다. 데이터를 꺼낼 때 우선 순위가 높은 데이터가 가장 먼저 나온다는 특징이 있어 많이 활용되고 있습니다.

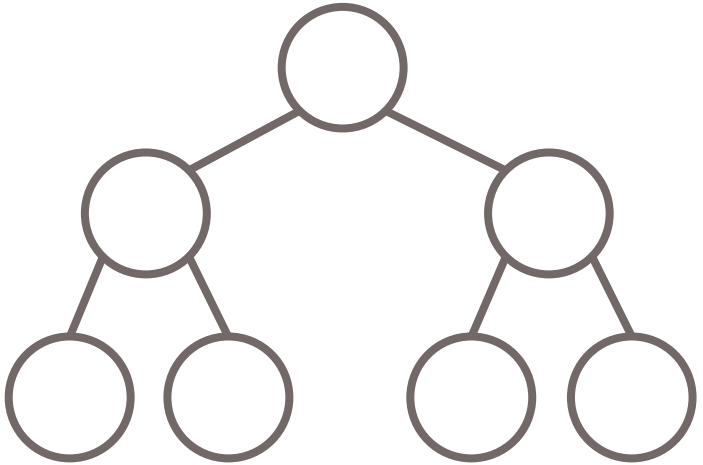
우선순위 큐는 운영체제의 작업 스케줄링, 정렬, 네트워크 관리 등의 다양한 기술에 적용되고 있습니다.

우선순위 큐

우선순위 큐와 큐의 차이점

일반적인 형태의 큐는 선형적인 형태를 가지고 있지만 우선순위 큐는 트리(Tree) 구조로 보는 것이 합리적입니다. 일반적으로 우선순위 큐는 최대 힙을 이용해 구현합니다.

일반적인 큐

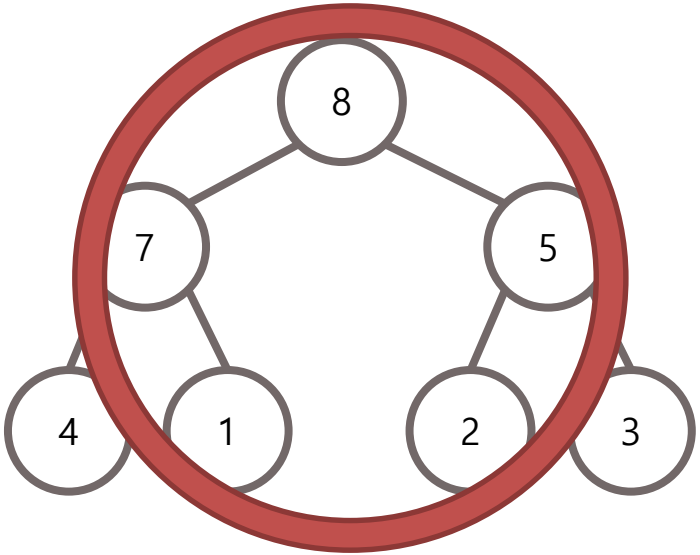


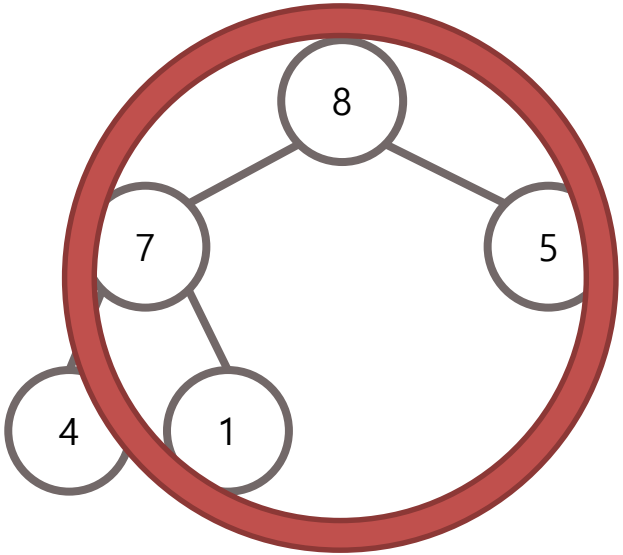
우선순위 큐

우선순위 큐

최대 힙 (Max Heap)

최대 힙은 부모 노드가 자식 노드보다 값이 큰 완전 이진 트리를 의미합니다.

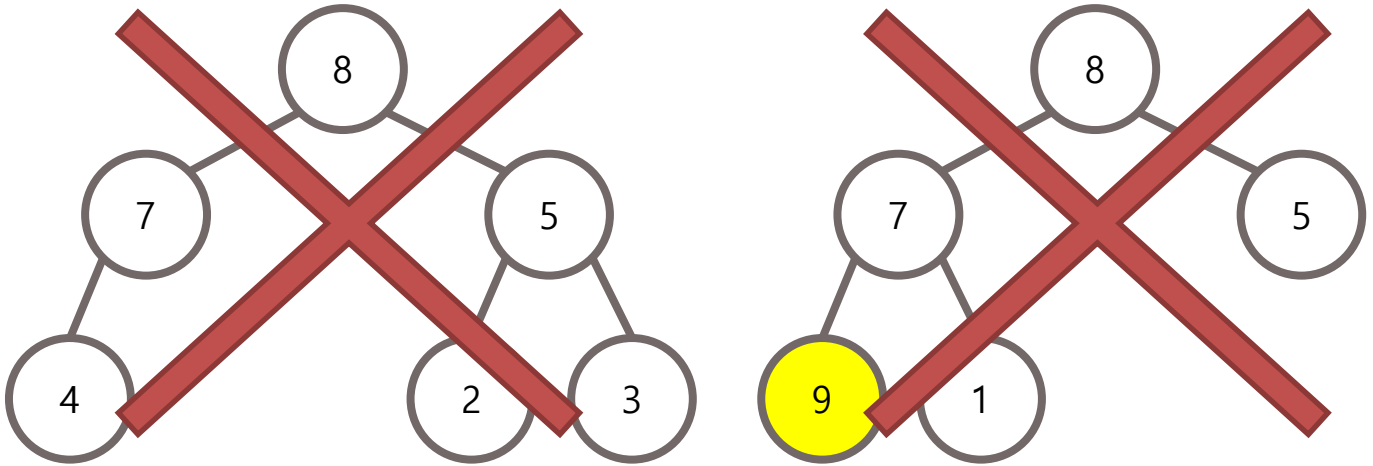




우선순위 큐

최대 힙 (Max Heap)

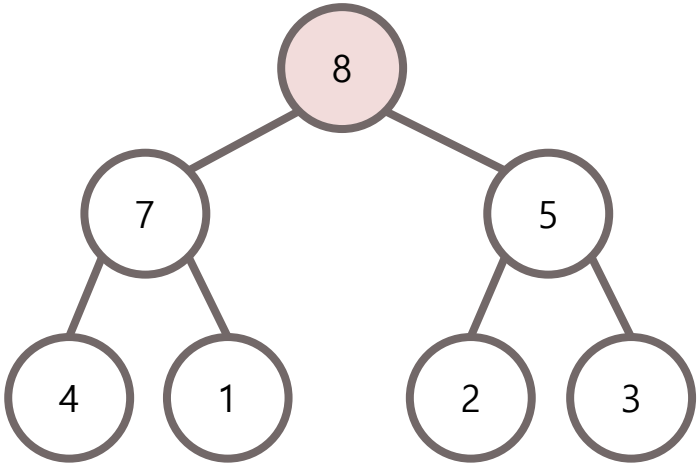
최대 힙은 부모 노드가 자식 노드보다 값이 큰 완전 이진 트리를 의미합니다.



우선순위 큐

최대 힙 (Max Heap)

최대 힙의 루트 노드는 전체 트리에서 가장 큰 값을 가진다는 특징이 있습니다.



우선순위 큐

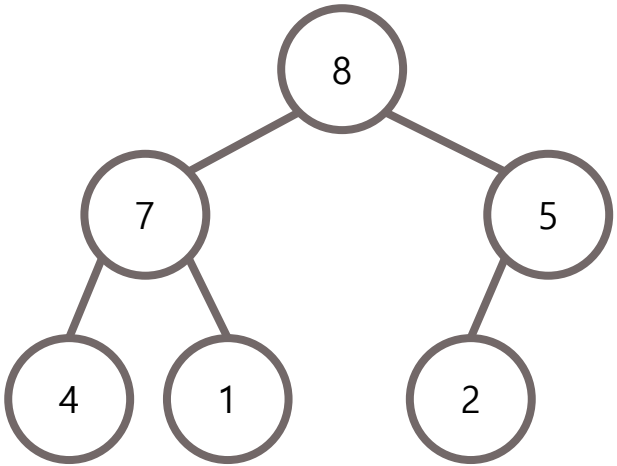
최대 힙 (Max Heap)

우리는 항상 전체 트리가 최대 힙 구조를 유지하도록 자료구조를 만들 수 있습니다.

- PUSH: 우선순위 큐에 데이터를 삽입합니다.
- POP: 우선순위 큐에서 데이터를 추출합니다.

우선순위 큐

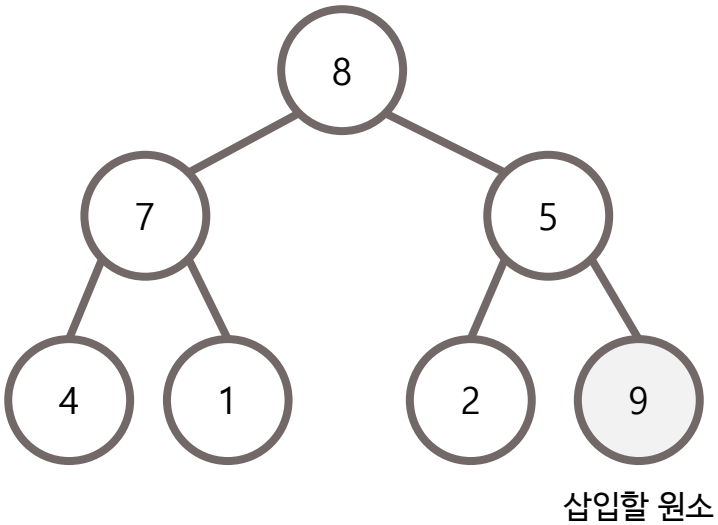
우선순위 큐의 삽입



우선순위 큐

우선순위 큐의 삽입

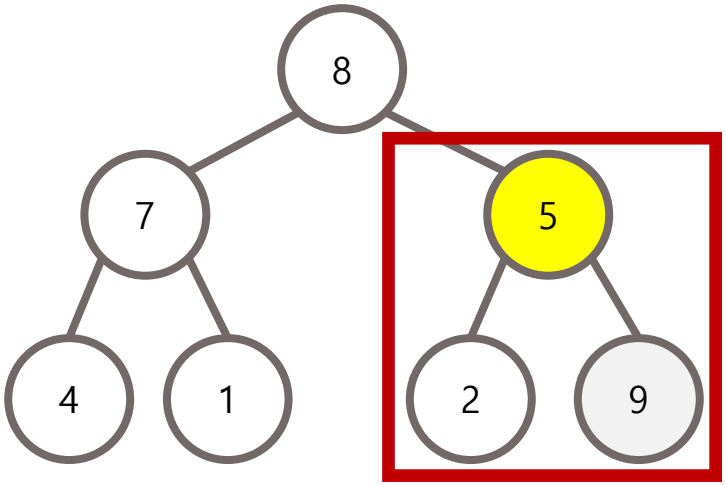
삽입할 원소는 완전 이진 트리를 유지하는 형태로 순차적으로 삽입됩니다.



우선순위 큐

우선순위 큐의 삽입

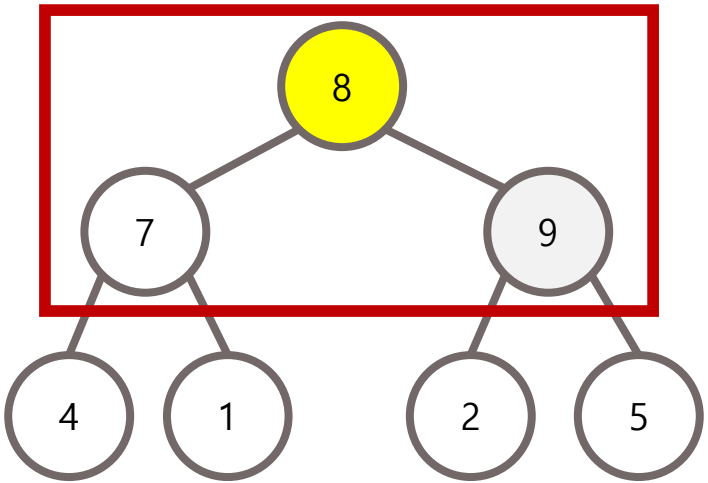
삽입 이후에는 루트 노드까지 거슬러 올라가면서 최대 힙을 구성합니다. **[상향식]**



우선순위 큐

우선순위 큐의 삽입

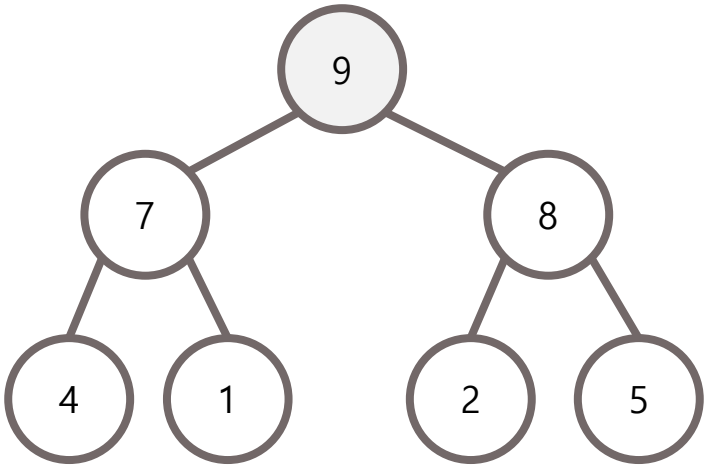
삽입 이후에는 루트 노드까지 거슬러 올라가면서 최대 힙을 구성합니다. **[상향식]**



우선순위 큐

우선순위 큐의 삽입

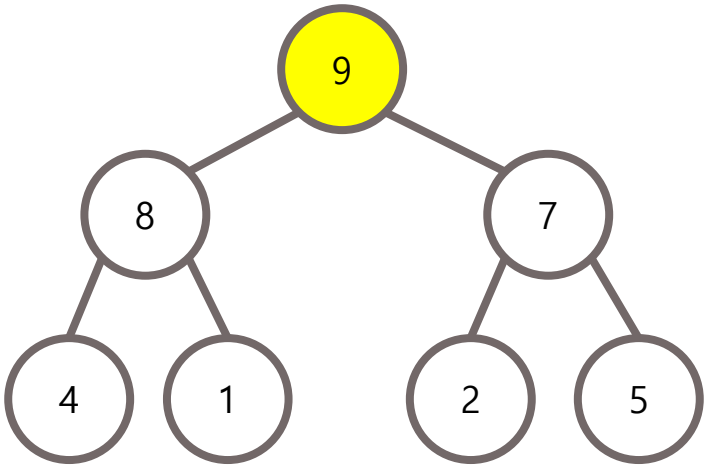
삽입 이후에는 루트 노드까지 거슬러 올라가면서 최대 힙을 구성합니다. **[상향식]**



우선순위 큐

우선순위 큐의 삭제

삭제를 할 때는 루트 노드를 삭제하고, 가장 마지막 원소를 루트 노드의 위치로 옮겨줍니다.

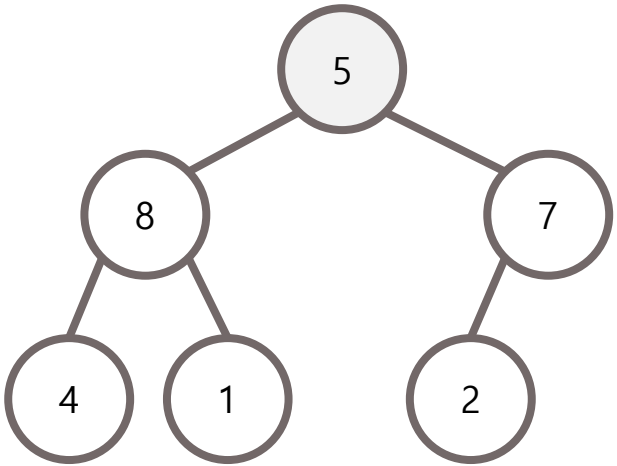


우선순위 큐

우선순위 큐의 삭제

삭제를 할 때는 루트 노드를 삭제하고, 가장 마지막 원소를 루트 노드의 위치로 옮겨줍니다.

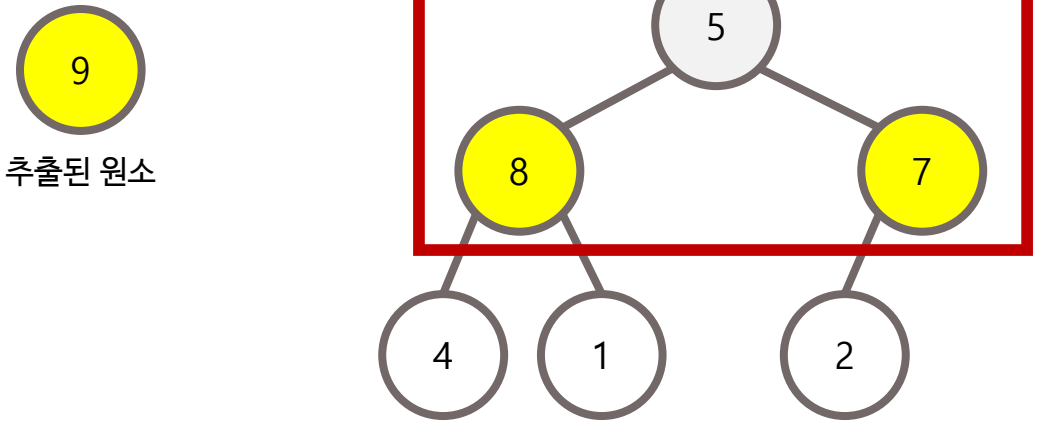
추출된 원소



우선순위 큐

우선순위 큐의 삭제

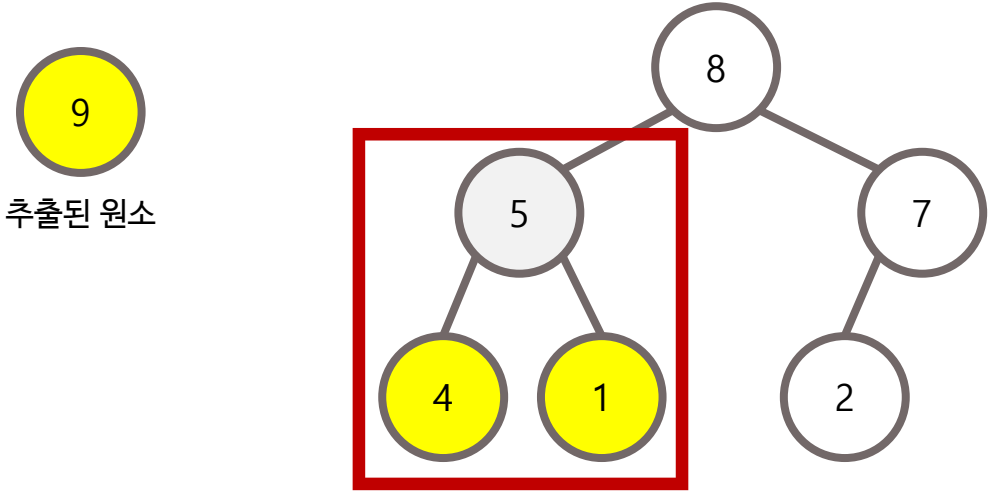
삭제 이후에는 리프 노드까지 내려가면서 최대 힙을 구성합니다. **[하향식]**



우선순위 큐

우선순위 큐의 삭제

삭제 이후에는 리프 노드까지 내려가면서 최대 힙을 구성합니다. **[하향식]**

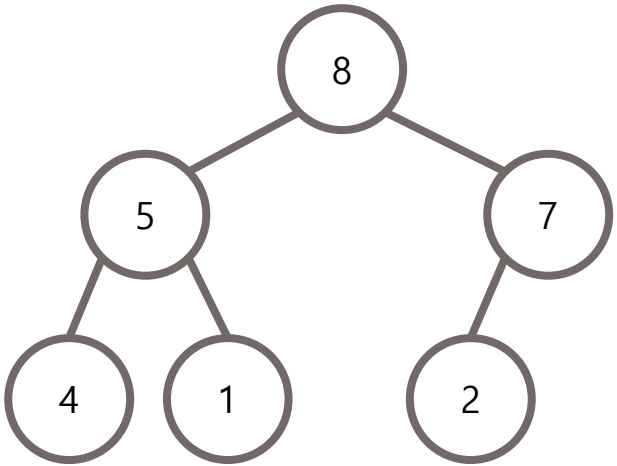


우선순위 큐

우선순위 큐의 삭제

삭제 이후에는 리프 노드까지 내려가면서 최대 힙을 구성합니다. **[하향식]**

추출된 원소



우선순위 큐

우선순위 큐의 정의

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define MAX_SIZE 10000

void swap(int *a, int *b) {
  int temp = *a;
  *a = *b;
  *b = temp;
}

typedef struct {
  int heap[MAX_SIZE];
  int count;
} priorityQueue;
```

우선순위 큐

우선순위 큐의 삽입

```
void push(priorityQueue *pq, int data) {
  if (pq->count >= MAX_SIZE) return;
  pq->heap[pq->count] = data;
  int now = pq->count;
  int parent = (pq->count - 1) / 2;
  // 새 원소를 삽입한 이후에 상향식으로 힙을 구성합니다.
  while (now > 0 && pq->heap[now] > pq->heap[parent]) {
    swap(&pq->heap[now], &pq->heap[parent]);
    now = parent;
    parent = (parent - 1) / 2;
  }
  pq->count++;
}
```

우선순위 큐

우선순위 큐의 추출

```
int pop(priorityQueue *pq) {
  if (pq->count <= 0) return -9999;
  int res = pq->heap[0];
  pq->count--;
  pq->heap[0] = pq->heap[pq->count];
  int now = 0, leftChild = 1, rightChild = 2;
  int target = now;
  // 새 원소를 추출한 이후에 하향식으로 힙을 구성합니다.
  while (leftChild < pq->count) {
    if (pq->heap[target] < pq->heap[leftChild]) target = leftChild;
    if (pq->heap[target] < pq->heap[rightChild] && rightChild < pq->count) target = rightChild;
    if (target == now) break; // 더 이상 내려가지 않아도 될 때 종료
    else {
      swap(&pq->heap[now], &pq->heap[target]);
      now = target;
      leftChild = now * 2 + 1;
      rightChild = now * 2 + 2;
    }
  }
  return res;
}
```

우선순위 큐

우선순위 큐의 사용

```
int main(void) {
  int n, data;
  scanf("%d", &n);
  priorityQueue pq;
  pq.count = 0;
  for (int i = 0; i < n; i++) {
    scanf("%d", &data);
    push(&pq, data);
  }
  for (int i = 0; i < n; i++) {
    int data = pop(&pq);
    printf("%d ", data);
  }
  system("pause");
  return 0;
}
```

배운 내용 정리하기

우선순위 큐

1) 우선순위 큐는 완전 이진 트리 형태의 힙을 이용해 구현할 수 있습니다.
2) 우선순위 큐의 삽입과 삭제는 $O(logN)$의 시간 복잡도를 가집니다.
3) 따라서 우선순위 큐를 이용한 정렬은 $O(NlogN)$의 시간 복잡도를 가집니다.