Max Heap을 이용한 레스토랑 시스템

중앙대학교 소프트웨어학과 20186889 권용한

●소스코드 설명

```
#define MAX_HEAP_SIZE 10    //heap size+1
#define MAX_QUEUE_SIZE 10   //queue size+1
#define HEAP_FULL(n) (n==MAX_HEAP_SIZE-1)
#define HEAP_EMPTY(n) (!n)

typedef struct element {
    int wait_num;
    int peo_num;
}element;

element heap[MAX_HEAP_SIZE];
element queue[MAX_QUEUE_SIZE];

int front = 0;
int rear = 0;
```

프로그램에서 사용되는 heap과 queue를 생성하는 코드입니다. 이때 heap의 대기번호인 wait_num을 key로 하고, 대기번호는 9부터 내림차순으로 부여되며, 1번 고객을 마지막으로 마지막 고객이 식사를 종료하거나 0모드가 선택된다면 프로그램을 종료하기 때문에, heap의 사이즈가 9까지만 사용되면 됩니다. 또한 heap의 사이즈가 9까지만 사용되기 때문에 queue에서도 최대 9까지만 queue에 add하게 됩니다. 때문에, heap과 queue의 최대 개수가 9를 넘지 않으므로 사이즈가 10인 배열로 선언합니다. heap연산 과정에서 배열의 0번째 원소부터 사용하는 것 보다 1번 원소부터 사용하는 것이 더 코드가 간결해지기 때문에 9가 아닌 10으로 선언합니다.

```
void heap_push(element item, int *n){
    int i;
    if (HEAP_FULL(*n)) {

    }
    i = ++(*n);
    while ((i != 1) && (item.wait_num > heap[i / 2].wait_num)) {
        heap[i] = heap[i / 2];
        i /= 2;
    }
    heap[i] = item;
}
```

heap의 연산 중 heap에 원소를 push하는 함수입니다. heap에 저장될 데이터인 element와 현재 heap에 남아있는 원소의 개수를 저장하는 변수의 주소를 입력받습니다. 이때 주소값을 입력 받는 이유는, heap에 push되고 나면 원소의 개수가 하나 증가하게 되는데, 값을 증가시키는 연산을 함수에서 진행하기 때문에 주소값으로 입력을 받습니다. 그리고 key값을 비교하여 heap을 max heap 조건에 맞도록 정렬하며 새로운 데이터를 heap에 추가하게 됩니다.

```
element heap_pop(int *n)
{
    int parent, child;
    element item, temp;
    if (HEAP_EMPTY(*n))
    {
        item.peo_num = 5;
        item.wait_num = 0;
        return item;
    }

    item = heap[1];
    //adjust
    temp = heap[(*n)--];
    parent = 1;
    child = 2;
    while (child <= *n) {
        if ((child < *n) && (heap[child].wait_num < heap[child + 1].wait_num))
            child++;
        if (temp.wait_num >= heap[child].wait_num)
            break;
        heap[parent] = heap[child];
        parent = child;
        child *= 2;
    }
    heap[parent] = temp;
    return item;
}
```

heap에서 pop하는 함수로, 이때 heap이 empty가 되는 상황에서 pop이 또 일어나는 경우가 main에서 일어날 수

있는데, heap이 pop이 일어나는 경우가 heap의 데이터 중 peo_num이 좌석의 수보다 큰 경우 다시 pop을 하게 됩니다. 그리고 좌석의 수보다 작거나 같은 인원이 있는 경우에만 좌석을 배정하고 그렇지 않은 경우엔 배정 불가를 출력하게 되는데 최대 좌석의 수가 4이기 때문에 heap이 empty인 상황에서 pop을 하였을 때, 4보다 큰 값인 5를 peo_num으로 가지는 item을 return하도록 하여 main에서 배정이 가능한 경우와 배정이 불가능한 경우를 나누면서 heap이 empty인 상황을 함께 알 수 있도록 하였습니다.

```
void addq(element item){
    if ((rear + 1) % MAX_QUEUE_SIZE == front) {
    }
    rear = (rear + 1) % MAX_QUEUE_SIZE;
    queue[rear] = item;
}
element deleteq() {
    element x;

    if (front == rear) {
        x.peo_num = 0;
        x.wait_num = 0;
        return x;
    }
    front = (front + 1) % MAX_QUEUE_SIZE;
    x = queue[front];
    return x;
}
```

프로그램이 heap에서 pop된 값들 중 좌석이 배정되지 않은 경우의 값들은 queue에 저장하고, 좌석 배정이 끝나거나 배정이 불가능하다고 판단이 된다면 queue에 저장된 값들을 다시 heap에 저장합니다. 이때 필요한 queue의 연산들입니다. 이때 다시 heap에 다시 값을 저장하기 위해 queue에서 delete를 하는 과정에서 queue가 empty인 상황이 온다면 peo_num이 0인 값을 가진 데이터를 return하는데 이를 통해 queue가 empty인 상황을 인지할 수 있도록 하였습니다.

```
if (select_func == 0)
{
    printf("End of Program₩n");
    return 0;
}
else if (select_func == 1)
{
    printf("대기번호:%d₩n", waitingNum);
    printf("인원수?:");
    scanf_s("%d", &people_num);
    customer.wait_num = waitingNum;
    customer.peo_num = people_num;
    heap_push(customer, &heapsize);
    waitingNum--;
    desk_wait_num++;
    printf("식탁 대기수: %d₩n", desk_wait_num);
    printf("----------₩n");

}
```

여기서 부터는 main함수의 코드로, 첫 번째 if문은 0번 기능이 선택된 경우, End of Program을 출력하고 프로그램을 종료하도록 하였으며, 1번 기능이 선택된 경우엔 현재의 대기 번호를 출력하고 인원수를 입력받으며, 입력받은 인원수와 대기번호를 데이터로 가진 element 구조체를 만들어 그 데이터를 heap에 push해 줍니다. 그리고 waitingNum을 1 감소시키면서 다음 waitingNum의 값을 조정해줍니다.

```
else if (select_func == 2)
{
    seat_assign_success = 1;
    printf("식탁 크기?");
    scanf_s("%d", &seat_size);
    do {
        customer = heap_pop(&heapsize);
        if (customer.peo_num > seat_size) {
            addq(customer);
            seat_size_success = 1;
            if (customer.peo_num == 5)
            {
                seat_assign_success = 0;
            }
        }
        else {
            seat_size_success = 0;
            desk_wait_num--;
        }
    } while (seat_size_success&&seat_assign_success);

    if (seat_size_success == 0)
    {
        printf("배정:: 대기번호: %d 인원수: %d₩n", customer.wait_num, customer.peo_num);
    }
    else {
        printf("배정불가::₩n");
    }

    while (1) {
        customer = deleteq();
        if ((customer.peo_num==0)||(customer.peo_num==5))
            break;
        heap_push(customer, &heapsize);
    }


    printf("식탁 대기수: %d₩n", desk_wait_num);
    printf("----------₩n");
}
```

2번 기능이 선택된 경우로, 식탁 사이즈를 입력받은 이후, 입력된 값보다 작거나 같은 수의 사람 수를 데이터로 가진 heap이 있을 때 까지 heap에서 데이터를 pop하고 pop된 데이터 중 식탁 사이즈보다 더 많은 사람의 수를 데이터로 가진 element를 queue에 저장합니다. 그리고 이 과정을 heap이 empty해지는 상황까지 반복할 수도 있기 때문에, loop를 heap이 empty인 상황과 좌석배정에 성공한 경우에 loop를 탈출하도록 하였습니다. 그리고 loop를 탈출한 이후 탈출한 원인이 좌석배정에 성공한 경우인지 아니면 배정에 실패한 경우인지를 판단하기 위해 seat_size_success가 0이면 좌석배정에 성공한 경우로 여기고, seat_size_success가 0이 아닌 경우엔 배정에 불가한 경우로 판단합니다. 이때 seat_assign_success는 좌석 배정에 성공하지 못하고 heap이 empty인 상황에서 loop를 탈출하기 위해 만들어진 변수입니다.

loop를 탈출하고 queue에 저장된 값들을 다시 heap에 push하는데, loop를 탈출할 때 1.좌석배정에 성공인 경우엔 peo_num이 0인 데이터를 받으며 탈출하고 2.좌석배정에 실패한 경우엔 peo_num이 5인 데이터를 받으며 탈출하게 됩니다. 때문에 peo_num이 0이나 5인 값은 실제 데이터가 아닌 일종의 dummy data로 판단할 수 있는데, queue에서 delete된 값들중 이 값들이 나오면 heap에 push하지 않고 loop를 탈출합니다.

●출력결과

```
Start of Program
기능:
0.종료 1.대기접수 2.식탁배정
식탁 대기수: 0
-------------
기능 선택?:1
대기번호:9
인원수?:3
식탁 대기수: 1
-------------
기능 선택?:1
대기번호:8
인원수?:4
식탁 대기수: 2
-------------
기능 선택?:1
대기번호:7
인원수?:1
식탁 대기수: 3
-------------
기능 선택?:2
식탁 크기?4
배정:: 대기번호: 9 인원수: 3
식탁 대기수: 2
-------------
기능 선택?:2
식탁 크기?2
배정:: 대기번호: 7 인원수: 1
식탁 대기수: 1
-------------
기능 선택?:1
대기번호:6
인원수?:3
식탁 대기수: 2
-------------
기능 선택?:2
식탁 크기?2
배정불가::
식탁 대기수: 2
-------------
기능 선택?:2
식탁 크기?4
배정:: 대기번호: 8 인원수: 4
식탁 대기수: 1
-------------
```

```
기능 선택?:1
대기번호:5
인원수?:2
식탁 대기수: 2
-------------
기능 선택?:2
식탁 크기?2
배정:: 대기번호: 5 인원수: 2
식탁 대기수: 1
-------------
기능 선택?:2
식탁 크기?2
배정불가::
식탁 대기수: 1
-------------
기능 선택?:2
식탁 크기?4
배정:: 대기번호: 6 인원수: 3
식탁 대기수: 0
-------------
기능 선택?:0
End of Program
계속하려면 아무 키나 누르십시오 . . .
```

pdf의 값들을 예시로 입력한 경우입니다.
마지막에 0이 입력되어 프로그램이 종료된 것을 확인할 수 있으며, 데이터가 저장된 것을 정리해 보자면 대기번호와 인원 순으로 9,3 8,4 7,1 6,3 5,2가 저장되고 식탁이 각각 4 2 2 4 2 2 4 순으로 자리가 났는데. 첫 4일 때는 9번 대기자가 좌석을 배정받았고, 그다음번 2 식탁이 자리가 났을땐, 8번은 좌석보다 인원이 더 많아서 7번이 배정을 받습니다. 그리고 다시 2번 자리가 났는데 이때는 6번까지밖에 입력이 되지 않았고, 때문에 좌석을 배정할 수 없어 배정이 불가함을 출력합니다. 그리고 4번 자리가 생겨 8번 손님이 배정을 받고 5번 손님이 대기하고 그 이후 2번 자리가 나서 5번 손님이 좌석을 배정받습니다. 그리고 4번 자리가 비어서 마지막 6번 손님이 배정을 받은 후 6번 손님을 끝으로 프로그램이 종료됩니다.

9,3 8,4 7,1 입력 후 4번자리가 생겨 9번이 배정받고 8,4 7,1이 힙에 존재합니다. 2번 자리가 생겨 7번이 배정받고 8,4가 힙에 남아있습니다. 그리고 6,3이 입력되고 2번자리가 나지만 힙에 있는 8번과 6번은 인원이 2보다 커서 배정을 받지 못합니다. 4번자리가 나와서 8번이 배정받고 6번은 힙에 남아있습니다. 그리고 5,2가 입력되고 2번 자리가나와 5번이 배정받습니다. 이후 다시 2번자리가 나오는데, 6번은 3명이여서 배정을 받지 못하고, 이후 4번 자리가 나와 6번이 좌석을 배정받습니다.

```
Start of Program
기능:
0.종료 1.대기접수 2.식탁배정
식탁 대기수: 0
-------------
기능 선택?:1
대기번호:9
인원수?:4
식탁 대기수: 1
-------------
기능 선택?:1
대기번호:8
인원수?:2
식탁 대기수: 2
-------------
기능 선택?:2
식탁 크기?4
배정:: 대기번호: 9 인원수: 4
식탁 대기수: 1
-------------
기능 선택?:1
대기번호:7
인원수?:3
식탁 대기수: 2
-------------
기능 선택?:1
대기번호:6
인원수?:1
식탁 대기수: 3
-------------
기능 선택?:
2
식탁 크기?2
배정:: 대기번호: 8 인원수: 2
식탁 대기수: 2
-------------
기능 선택?:1
대기번호:5
인원수?:3
식탁 대기수: 3
-------------
기능 선택?:2
식탁 크기?2
배정:: 대기번호: 6 인원수: 1
식탁 대기수: 2
-------------
```

```
기능 선택?:2
식탁 크기?4
배정:: 대기번호: 7 인원수: 3
식탁 대기수: 1
-------------
기능 선택?:1
대기번호:4
인원수?:4
식탁 대기수: 2
-------------
기능 선택?:1
대기번호:3
인원수?:1
식탁 대기수: 3
-------------
기능 선택?:2
식탁 크기?2
배정:: 대기번호: 3 인원수: 1
식탁 대기수: 2
-------------
기능 선택?:2
식탁 크기?4
배정:: 대기번호: 5 인원수: 3
식탁 대기수: 1
-------------
기능 선택?:1
대기번호:2
인원수?:2
식탁 대기수: 2
-------------
기능 선택?:1
대기번호:1
인원수?:3
식탁 대기수: 3
-------------
기능 선택?:2
식탁 크기?2
배정:: 대기번호: 2 인원수: 2
식탁 대기수: 2
-------------
기능 선택?:2
식탁 크기?4
배정:: 대기번호: 4 인원수: 4
식탁 대기수: 1
```

```
기능 선택?:2
식탁 크기?2
배정불가::
식탁 대기수: 1
-------------
기능 선택?:2
식탁 크기?4
배정:: 대기번호: 1 인원수: 3
식탁 대기수: 0
-------------
계속하려면 아무 키나 누르십시오 . . .
```

이번엔 마지막으로 1번 손님까지 입력받고 1번 손님이 좌석 배정을 받고 식사를 마쳐서 프로그램이 종료되었습니다. 입력 받은 순서부터 각각 9,4 8,2 7,3 6,1 5,3 4,4 3,1 2,2 1,3 이고, 자리가 각각 4 2 2 4 2 4 2 4 2 4 순으로 자리가 났는데 이때 첫 4때 9번이 배정받고, 2에 8번이 배정받았습니다. 그 다음 2에는 7번은 배정받지 못하고 6번이 배정 받으며 그 후 4번자리가 나와 7번이 배정을 받았습니다. 그리고 각각 2 4번 자리가 나와서 3번과 5번 손님이 자리를 배정받고 다시 2 4번 자리가 나와 이번에는 2번 손님, 4번손님 순으로 자리를 배정 받았습니다. 마지막으로 2번과 4번 자리가 나왔는데, 1번 손님은 인원이 3명이기에 2번 자리가 나왔을땐 배정받지 못하고 4번자리가 나와서 배정을 받은 후 프로그램이 종료하게 되었습니다.

9,4 8,2가 입력되고, 4번자리가 나와 9번이 배정받습니다. 그리고 7,3 6,1이 입력됩니다. 그리고 2번 자리가 나와 인원수가 2인 8번 손님이 배정을 받고 힙에는 7번과 6번 손님이 남아있습니다. 그리고 5,3이 입력되고 2번자리가 나와서 6번손님이 입장합니다. 그리고 4번 자리가 놔와 이제 7번 손님이 입장합니다. 4,4 3,1가 입력되어 5번과 4번이 대기중이고 이후 2번자리가 나와서 인원수가 2보다 더 큰 5번과 4번은 배정받지 못하고 3번이 배정받게 됩니다. 그리고 4번자리가 나와서 5번 손님이 입장합니다. 그리고 2,2 1,3이 입력되어 현재 4 2 1번 손님이 대기중이고 이후 2번자리가 나와서 2번 손님이 입장하고, 4번자리가 나와 4번 손님도 입장합니다. 그리고 2번 자리가 다시 나왔으나 1번은 3명이라 배정받지 못하고 이후 4번 자리가 나와 자리를 배정받고, 1번 손님까지 식당에 입장하고 대기수가 없기 때문에 프로그램이 종료되게 됩니다.