컴퓨터공학실험1 7주차 예비보고서

전공: 컴퓨터공학과 학년: 2 학번: 20191559 이름: 강상원

1. 2주차 실습에 구현하는 랭킹 시스템에 대한 자료를 읽어보고, 이를 구현하기 위한 자료구조를 2가지 이상 생각한다.

   - Linked List

     테트리스 게임이 정상적으로 종료될 때 (Game Over) 이름을 입력받고, 이에 따라 이름과 점수가 저장된 노드를 추가 생성한다. 링크드 리스트에서 점수가 가장 큰 노드부터 시작해 내림차순으로 정렬이 된다. rank에 입력이 0명인 경우, 1명인 경우, n명 (n≥2)인 경우로 나누어 링크를 다라 대소를 비교하며 알맞은 위치에 새 노드를 집어넣는다. 전체 노드 개수가 n개일 때, 시간 복잡도를 계산하자면, 최악의 경우 O(n)이다. (점수가 젤 작으면 끝까지 탐색해야 한다.) 공간복잡도 또한 O(n)이다.

   - Simple Array (continuous)

     링크드 리스트가 아닌, 연속적인 배열 형태로 rank를 저장한다. 이 때, 삽입과 삭제 시 그 뒤의 원소들을 모두 한 칸씩 움직여야 하는 비효율성이 있다. 최악의 경우 시간복잡도, 공간복잡도는 모두 O(n)이 나온다.

     연산 횟수의 효율성에 있어서, 연속적 단순 배열 형태는 중간에 삽입과 삭제가 번거로우므로 Linked List가 더 적합한 자료구조라 생각한다.

2. 생각한 각 자료구조에 대해서 새로운 랭킹을 삽입 및 삭제를 구현하기 위한 pseudo code를 작성하고, 시간 및 공간 복잡도를 계산한다.

   - Linked List

```
insert_node(name, score)
  Make new node* new
  new->name = name
  new->score = score
  new->link = NULL
  head_node.n++

  if there is no node
        head_node.HEAD = new
  else if there is 1 node
        if new->score > head_node.HEAD->score
```

```
                new->link = head_node.HEAD
                head_node.HEAD = new
        else
                head->node.HEAD->link = new
  else (if there are n≥2 nodes)
        if new->score > head_node.HEAD->score
                new->link = head_node.HEAD
                head_node.HEAD = new
                return
  else
        iterate until… (new->score > idxx->score) || (idxx->link == NULL)
                front = idxx
                idxx = idxx->link
        if (front->score > new->score) && (new->score > idxx->score)
                front->link = new
                new->link = idxx
        else
                idxx->link = new
```

최악의 경우 시간복잡도는 앞서 기술하였듯이 O(n)이다.

```
delete_node(n)
        idx = HEAD

        if there is 1 node
                remove idx node
        else
                for i=1 to i=n-1
                        if idx==NULL
                                error
                        else
                                idx=idx->link
                        back=idx.link
                        idx.link=back.link
                        Remove back node
```

n번째 원소까지 순차적으로 탐색하므로 시간 복잡도는 O(n)이다.

(공간 복잡도도 마찬가지)

- Simple Array (continuous)

```
insert_node(name, score)
  idx=0
  while (arr[idx].score ≥ score)
        idx++
        break
  for i=len to idx
        arr[i+1].name = arr[i].name
        arr[i+1].score = arr[i].score
  arr[idx].name = name
  arr[idx].score = score
  len++
```

최악의 경우 시간복잡도: O(n)

```
delete_node(x)
  for i= x to len
        arr[i].name = arr[i+1].name
        arr[i].score = arr[i+1].score
  len--
```

시간복잡도는 O(n)이다.

3. 생각한 각 자료구조에서 사용자가 부분적으로 확인하길 원하는 정렬된 랭킹(x~y위, x≤y, x, y는 정수)의 정보를 얻는 방법을 간략히 요약해서 pseudo code로 작성하고, 시간 및 공간 복잡도를 계산한다.

- Linked List

```
input X, Y
Node* idx = head_node.HEAD
index=1
while (1) (iterate)
        if idx==NULL
              break;
        if X≤index && index≤Y
              print name, score in index
        index++
        idx = idx->link
```

시간 복잡도는 O(Y-X) (O(1), 상수 시간)이다. 공간 복잡도도 마찬가지이다.

- Simple Array (continuous)

```
input X, Y
for i=X to Y
      print arr[i].name, arr[i].score
```

시간 복잡도는 O(Y-X) (O(1), 상수 시간)이다. 공간 복잡도도 마찬가지이다.