전공 : 컴퓨터공학과 학년 : 4 학번 : 20212020 이름 : 박민준

**1. 실험시간에 작성한 랭킹 시스템의 자료구조와 랭킹 시스템의 각 기능에 대한 알고리즘을 요약하여 기술하시오. 본인이 선택한 랭킹 시스템을 구현하기 위한 자료구조가 왜 효율적인지 시간 및 공간 복잡도를 통해 보이고, 설명하시오.**

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

- 랭킹 시스템을 구현하기 위해 Linked List (연결 리스트) 자료구조를 사용하였다. Linked List를 구현하기 전에, 먼저, Node 구조체를 만들었다. name과 score는 각각 사용자의 이름과 점수를 저장하는 변수이며 next는 다음 Node를 가리키는 포인터 변수이다.

1) createRankList() 함수

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

- createRankList() 함수는 “rank.txt” 파일로부터 랭킹 정보를 읽어와 연결 리스트를 생성하는 함수이다.

- file pointer에 해당하는 fp 변수를 이용해 파일로부터 랭킹 정보를 읽어온다. 만약 파일이 존재하지 않으면 빈 리스트를 유지한다.

- 파일을 열고 랭킹의 수를 읽어온 후, 랭킹의 수만큼 반복문을 돌며 각 랭킹 정보를 읽어 Node를 생성하고 연결 리스트에 추가하는 작업을 진행한다.

- 만약 head가 NULL값을 가진다면 비어 있는 리스트를 의미하므로 newNode를 head로 설정해준다.

- 연결 리스트에 랭킹 정보를 추가할 때는 점수 순으로 내림차순 정렬되도록 삽입한다.

- 해당 함수는 파일에서 모든 랭킹 정보를 읽어와 연결 리스트를 생성하기 때문에 O(n)의 시간 복잡도를 갖는다. 여기서 n은 랭킹의 수를 의미한다.

2) rank() 함수

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

- rank() 함수는 사용자가 입력한 범위에 해당하는 랭킹 정보를 출력하는 함수이다.

- 먼저, 사용자로부터 X와 Y의 값을 입력 받는다. 그 후 연결 리스트의 head부터 순회하며 X부터 Y 범위까지 랭킹 정보를 출력한다. 즉, 현재 Node의 순위가 X보다 크거나 같고 Y보다 작거나 같으면 해당 Node의 정보를 출력한다.

- 만약 X와 Y의 값이 유효하지 않은 경우 예외 메시지를 출력한다.

- 해당 함수 또한 연결 리스트를 순회하며 X부터 Y 범위까지의 랭킹을 출력하기 때문에 O(n)의 시간 복잡도를 갖는다.

3) newRank() 함수

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

- newRank() 함수는 새로운 랭킹 정보를 연결 리스트에 추가하는 함수이다.

- 게임 종료 시, 사용자의 이름과 점수를 입력 받아 새로운 Node를 생성한다. 그 후 연결 리스트를 순회하며 새로운 Node를 점수 순으로 올바른 위치에 삽입한다.

- 만약 prev가 NULL값을 가진다면 newNode의 랭킹이 제일 높다는 것을 의미하므로 newNode를 연결 리스트의 맨 앞에 삽입하고 head로 설정해준다.

- 마지막에 랭킹의 수를 증가시키고, writeRankFile() 함수를 호출하여 파일에 새로운 랭킹 정보를 기록한다.

- 해당 함수는 최악의 경우, newNode를 연결 리스트의 맨 끝에 삽입해야 하므로 O(n)의 시간 복잡도를 갖는다.

4) writeRankFile() 함수

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

- 해당 함수는 현재 랭킹 정보를 “rank.txt” 파일에 기록하는 함수이다.

- 먼저 file pointer에 해당하는 fp 변수를 이용해 파일을 열고 랭킹의 수를 기록한다. 그 후, 반복문을 통해 연결 리스트의 head부터 시작하여 끝까지 각 Node의 정보를 파일에 기록한다.

- 해당 함수는 연결 리스트의 모든 Node를 파일에 기록해야 하기 때문에 O(n)의 시간 복잡도를 갖는다.

- 연결 리스트는 새로운 Node를 삽입할 때 적절한 위치를 찾기 위해 리스트를 순회해야 하므로 O(n)의 시간 복잡도를 갖는다.

- 하지만, 연결 리스트가 정렬된 상태로 유지되므로 특정 범위의 랭킹을 조회하거나 출력하는 작업 역시 O(n)의 시간 복잡도로 효율적이다.

- 공간 복잡도 측면에서, 각 노드는 사용자의 이름과 점수를 저장하므로 O(n)의 공간이 소요된다. 연결 리스트의 각 Node들은 고정된 크기의 메모리를 사용하므로 효율적이고, 추가적인 오버헤드가 적다.

- 즉, 전체적인 시간 및 공간 복잡도 측면에서 연결 리스트는 랭킹 정보를 저장하고 관리하는 데 적합한 자료구조이다.

**2. 본 실험 및 숙제를 통해 습득한 내용을 기술하시오.**

- 이번 실험 및 과제를 통해 랭킹 시스템을 구현하면서 Linked List 자료구조의 이해와 활용에 대해 깊이 있는 학습을 할 수 있었다. Node 구조체는 Linked List의 기본 단위로, 사용자의 이름과 점수를 저장하는 name과 score 변수, 그리고 다음 Node를 가리키는 포인터 변수 next로 구성된다. 이를 통해 Linked List의 각 Node가 다른 Node와 연결될 수 있으며, 데이터를 동적으로 관리할 수 있음을 배웠다.

- createRankList() 함수에서, 파일 입출력을 통해 데이터를 읽어와 동적으로 Node를 생성하고 이를 Linked List에 추가하는 방법을 학습하였다. 특히, 데이터가 점수 순으로 내림차순 정렬되도록 삽입하는 알고리즘을 구현하면서 Linked List의 동작 원리를 이해할 수 있었다.

- newRank() 함수를 통해 새로운 랭킹 정보를 Linked List에 추가하는 과정을 이해하였다. 새로운 Node를 생성하고, 이를 점수 순으로 올바른 위치에 삽입하는 과정을 구현하면서 Linked List의 삽입 알고리즘을 학습할 수 있었다. 또한, 새로운 Node가 리스트의 맨 앞이나 중간, 끝에 삽입되는 다양한 경우에 대하여 이를 처리하는 방법을 익힐 수 있었다.

- Linked List의 각 연산에 대한 시간 복잡도와 공간 복잡도를 Array 배열 등 다른 자료구조와 비교 분석하면서 자료구조의 효율성에 대해 다시 생각해볼 수 있었다. Node 삽입, 삭제, 검색 등의 연산이 O(n)의 시간 복잡도를 가지는 이유와 이를 최적화할 수 있는 방법에 대해 고민해볼 수 있었다. 마지막으로, Linked List가 정렬된 상태로 유지되므로 특정 범위의 랭킹을 조회하거나 출력하는 작업의 효율성에 대해 이해할 수 있었다.

**3. 과제 모드 1**

**1) 사용자의 이름을 입력하고 해당하는 랭킹 정보 출력 화면 첨부**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**2) 사용자 이름으로 원하는 사용자의 이름을 검색할 때의 시간 및 공간 복잡도**

- Linked List에서 특정 사용자의 이름을 검색하는 경우, 연결 리스트의 처음부터 끝까지 순회하며 각 Node의 name 값을 비교해야 한다. 검색하려는 사용자가 리스트의 마지막에 위치하거나 리스트에 존재하지 않는 최악의 경우, 리스트의 모든 Node를 확인해야 한다. 이 경우 시간 복잡도는 O(n)이 된다. 여기서 n은 Linked List의 Node 개수이다.

- Linked List에서 특정 사용자를 검색하는 경우, 검색 과정에서 별도의 데이터 구조를 사용하지 않기 때문에 추가적인 공간을 사용하지 않는다. 따라서 공간 복잡도는 O(1)이 된다.

**4. 과제 모드 2 – 선택한 자료구조에 맞춰 삭제 알고리즘을 그림으로 표현한다.**

도표, 텍스트, 기술 도면, 라인이(가) 표시된 사진

자동 생성된 설명