9주차 예비보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20191612 이름: 윤기웅

**1. (문제 및 제목은 작성하지 말 것. 답안만 작성하면 됩니다.)**

.................

1. 연결 리스트

연결 리스트는 노드들이 포인터를 통해 연결된 구조입니다. 각 노드는 사용자의 이름과 점수를 저장하며, 이전 노드와 다음 노드를 가리키는 prev와 next 포인터를 갖습니다. 새로운 데이터를 삽입할 때는 next 포인터를 사용해 연결된 노드들을 탐색하며, 점수들이 내림차순이 유지되도록 적절한 자리에 노드를 넣습니다. 데이터를 삭제할 때는 prev 또는 next 포인터를 통해 해당 노드를 찾아 삭제하고, 삭제된 노드의 prev 노드와 next 노드를 연결해 리스트의 연속성을 유지합니다.

1. 이진 탐색 트리(BST)

이진 탐색 트리(BST)는 각 노드가 최대 두 개의 자식 노드를 가질 수 있는 구조로, 부모 노드보다 작은 값을 왼쪽 자식 노드에, 큰 값을 오른쪽 자식 노드에 저장합니다. 노드는 사용자 정보를 저장하며, 삽입 시 트리를 탐색해 점수가 더 작으면 왼쪽에, 더 크면 오른쪽에 노드를 삽입합니다. 제거할 노드를 찾을 때는 점수를 비교하고, 삭제 후에는 BST의 구조를 유지하도록 제거할 노드의 위치를 조정합니다. 데이터의 동적 관리에 유용한 두 자료 구조는 각각의 장점과 사용 사례에 따라 선택적으로 사용됩니다.

................

**2.**

1) 연결리스트 : 탐색의 경우는 처음부터 마지막까지 찾아야 하니까 시간 복잡도와 공간 복잡도 모두 O(n)이 된다.

\* 삽입 :

함수 insertNode(헤드, 새 노드):

만약 헤드가 NULL이라면:

헤드 = 새 노드

반환

현재 = 헤드

이전 = NULL

반복 (현재 != NULL 그리고 현재의 점수 < 새 노드의 점수):

이전 = 현재

현재 = 현재의 다음

새 노드의 다음 = 현재

만약 이전이 NULL이 아니라면:

이전의 다음 = 새 노드

아니라면:

헤드 = 새 노드

\* 삭제 :

함수 deleteNode(헤드, 삭제할 노드):

현재 = 헤드

이전 = NULL

반복 (현재 != NULL 그리고 현재 != 삭제할 노드):

이전 = 현재

현재 = 현재의 다음

만약 현재 == NULL:

반환 // 노드를 찾지 못함

만약 이전 != NULL:

이전의 다음 = 현재의 다음

아니라면:

헤드 = 현재의 다음

2) 이진 탐색 트리 : 탐색할 때 트리의 높이만큼 수행하니 O(logN)이고 공간 복잡도는 O(1)이 된다. 노드 제거의 경우에도 탐색과 마찬가지로 O(logN)의 시간 복잡도와 O(1)의 공간 복잡도가 생긴다.

\* 삽입 :

함수 insertNode(루트, 새 노드):

만약 루트 == NULL:

루트 = 새 노드

반환 루트

만약 새 노드의 점수 < 루트의 점수:

루트의 왼쪽 = insertNode(루트의 왼쪽, 새 노드)

아니라면:

루트의 오른쪽 = insertNode(루트의 오른쪽, 새 노드)

반환 루트

\* 삭제:

함수 deleteNode(루트, 점수):

만약 루트 == NULL:

반환 NULL

만약 점수 < 루트의 점수:

루트의 왼쪽 = deleteNode(루트의 왼쪽, 점수)

아니면 만약 점수 > 루트의 점수:

루트의 오른쪽 = deleteNode(루트의 오른쪽, 점수)

아니라면:

만약 루트의 왼쪽 == NULL:

임시 = 루트의 오른쪽

반환 임시

아니면 만약 루트의 오른쪽 == NULL:

임시 = 루트의 왼쪽

반환 임시

임시 = 최소값노드(루트의 오른쪽)

루트의 점수 = 임시의 점수

루트의 오른쪽 = deleteNode(루트의 오른쪽, 임시의 점수)

반환 루트

함수 최소값노드(노드):

현재 = 노드

반복 (현재의 왼쪽 != NULL):

현재 = 현재의 왼쪽

반환 현재

3

연결리스트 사용 시 : 시간 복잡도는 O(n)이 되고 최악의 상황에는 리스트 전체를 순회해야 합니다. 그리고 공간 복잡도는 O(y-x+1)이라 O(1)입니다.

함수 getRankRange(head, x, y):

초기화: rank = 1

초기화: current = head

초기화: result = 빈 배열

// 리스트를 순회하면서 x위부터 y위까지의 요소를 찾습니다.

반복 (current가 NULL이 아니고 rank가 y보다 작거나 같을 때):

만약 rank가 x보다 크거나 같다면:

result에 current의 데이터 추가

current를 다음 노드로 이동

rank 증가

result 반환

이진 탐색 트리의 경우 : 시간 복잡도는 O(logn)이고 공간 복잡도는 O(y-x+1)이라 O(1)이 됩니다.

함수 getRankRange(root, x, y):

초기화: result = 빈 배열

초기화: rank = 0

초기화: stack = 빈 스택

초기화: current = root

// 중위 순회를 이용해 x위부터 y위까지의 요소를 찾습니다.

반복 (current가 NULL이 아니거나 스택이 비어있지 않을 때):

반복 (current가 NULL이 아닐 때):

current를 스택에 푸시

current를 current의 왼쪽 자식으로 설정

current에 스택에서 팝한 요소를 설정

rank 증가

만약 rank가 x보다 크거나 같고 y보다 작거나 같다면:

result에 current의 데이터 추가

만약 rank가 y보다 크다면:

반복 중단

current를 current의 오른쪽 자식으로 설정

result 반환