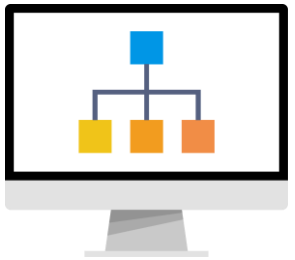


Data Structure & Algorithm
자료구조 및 알고리즘

19. 과제 2 및 실습 1 (Lab Session 1)

17강 보고서 돌아보기

- 오름차순으로 정렬된 배열이 있다. 이 배열에서 무작위 인덱스의 값 하나를 “아주 큰 수”로 덮어썼다.

- 이 배열은 “아주 큰 수” 하나를 제외하고는 거의 정렬되어 있는 상태인데 완벽히 정렬하고 싶다. 버블 정렬, 선택 정렬, 삽입 정렬 중 어떤 정렬 기법이 가장 효과적일까?
  - **삽입 정렬**

17강 보고서 돌아보기

- Quick sort + insertion sort 가 좋은 성능을 보이는 이유!

```
void BubbleSort(int arr[], int n)
{
    int i, j;
    int temp;

    for(i=0; i<n-1; i++)
    {
        for(j=0; j<(n-i)-1; j++)
        {
            if(arr[j] > arr[j+1])
            {
                // 데이터의 교환 /////
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

```
void SelSort(int arr[], int n)
{
    int i, j;
    int maxIdx;
    int temp;

    for(i=0; i<n-1; i++)
    {
        maxIdx = i;

        for(j=i+1; j<n; j++)     // 최솟값 탐색
        {
            if(arr[j] < arr[maxIdx])
                maxIdx = j;
        }

        // 교 환 ///////
        temp = arr[i];
        arr[i] = arr[maxIdx];
        arr[maxIdx] = temp;
    }
}
```

```
void InserSort(int arr[], int n)
{
    int i, j;
    int insData;

    for(i=1; i<n; i++)
    {
        insData = arr[i];

        for(j=i-1; j>=0 ; j--)
        {
            if(arr[j] > insData)
                arr[j+1] = arr[j];
            else
                break;
        }

        arr[j+1] = insData;
    }
}
```

18강 보고서 돌아보기

- 정렬의 성능 비교에는 최악의 경우 시간복잡도 외에도 여러 기준이 있다. 각 기준이 무엇을 뜻하는지 조사해보세요.
  - In-place / Out-place 정렬의 차이는 무엇일까?
  - Stable / Unstable 정렬의 차이는 무엇일까?

18강 보고서 돌아보기

- In-place 정렬: 추가적으로 필요한 메모리가 공간복잡도 O(1) 정렬

- Out-place 정렬: 그렇지 않은 정렬
  - 우리가 배운 힙 정렬의 경우 out-place지만 in-place로 개선 가능하다.
  - 병합 정렬의 경우 역시 out-place지만 이 in-place로 개선 가능하다. 단, 이 경우는 시간복잡도가 $O(N^2)$이 된다.

18강 보고서 돌아보기

- Stable 정렬: 같은 값을 가지는 데이터들이 정렬 후에도 입력 배열에서의 순서를 유지하는 정렬

- Unstable 정렬: 그렇지 않은 정렬
  - 데이터 간의 순서를 임의로 바꾸는 연산이 있을 때
  - 우리가 배운 정렬 중 선택 정렬, 힙 정렬, 퀵 정렬

- 많은 경우에
  - 시간을 희생해서 in-place로 만들거나
  - 공간을 희생해서 stable하게 만들 수 있다.

과제 2 소개

목표

하나의 이진 트리에서 노드의 삽입, 삭제, 순회를 수행하는 프로그램을 작성한다. 주어진 뼈대 코드(main.c)를 활용한다.

주의사항

- 프로그램의 초기에는 트리에 루트 노드 하나만 존재한다. 루트 노드에는 데이터 0 이 저장되어 있다. 루트 노드에 대한 삭제 명령어는 주어지지 않는다.
- 모든 명령어는 알파벳 소문자로 주어지며 트리에 저장하는 데이터는 항상 4 바이트 정수형 데이터이다. 따라서, 아래 명세에서 <data> 값은 항상 4 바이트 정수형 자료형으로 입력 및 표현할 수 있다.
- 아래 예제 명세에서 😊 기호의 왼편은 예제 입력, 오른편은 예제 입력에 대한 예제 출력을 나타낸다.
- 한 명령어에 대한 출력이 끝나면 줄 바꿈 문자를 출력하여 다음 명령어의 입력이 콘솔의 맨 왼쪽 끝에서 이루어질 수 있도록 하라.

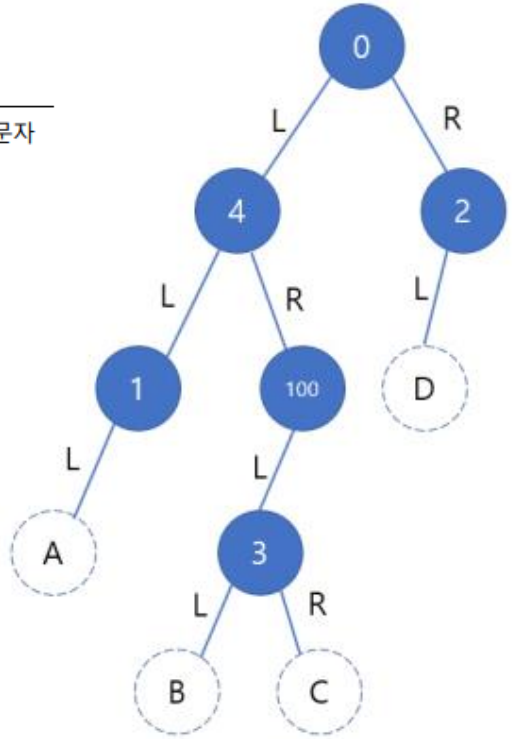
명세

프로그램은 표준 입력(scanf 를 이용)을 통하여 사용자에게 명령어를 입력 받고, 결과를 표준 출력(printf 를 이용)으로 출력한다. 종료 명령어가 입력될 때까지 반복적으로 명령어를 입력받아 처리해야 한다. 프로그램에서 지원하는 명령어는 아래와 같다.

add <pos_string> <data>

동작 트리의 <pos_string>위치에 <data> 값을 저장하는 노드를 추가한다. <pos_string>은 대문자 'L' 또는 'R'로 이루어진 길이 1 이상 20 이하의 문자열이며, 루트 노드로부터 시작하여 'L' 이면 왼쪽 자식 노드로 'R' 이면 오른쪽 자식 노드로 이동하여 <pos_string>이 끝나는 위치에 노드를 추가한다.

노드를 6 개 가지고 있는 아래 트리를 예로 들면,



위 트리에서 채워진 실선 노드만이 현재 트리에 존재하는 노드이고 점선 노드는 아래 예제를 위해 표시하였다.

"add LRLL 8"을 수행하면 B 위치에 데이터 8 을 가진 노드가 추가된다.

"add LRLR 8"을 수행하면 C 위치에 데이터 8 을 가진 노드가 추가된다.

"add RL 8"을 수행하면 D 위치에 데이터 8 을 가진 노드가 추가된다.

"add R 8"은 유효하지 않은 명령어이다. 이미 데이터 2 를 가지는 노드가 추가하고자 하는 자리에 존재하기 때문이다.

"add LLLL 8"은 유효하지 않은 명령어이다. A 위치에 노드가 있어야 A 위치에 있는 노드의 왼쪽 자식으로 데이터 8 을 가지는 새로운 노드를 추가할 수 있다. 현재 트리에서는 A 위치에 노드가 존재하지 않으므로 A 위치에 노드를 먼저 추가하고 위의 명령어를 수행하여야 한다.

add 명세

출력 추가에 성공하면 1 을 출력한다. 만약, <pos_string> 위치에 이미 노드가 존재하거나, <pos_string>을 따라가다 존재하지 않는 노드를 만날 경우에는 -1 을 출력한다.

예제 (현재 트리의 상태가 위 예와 같다면)

add LRLL 8 😊 1

add LRLR 8 😊 1

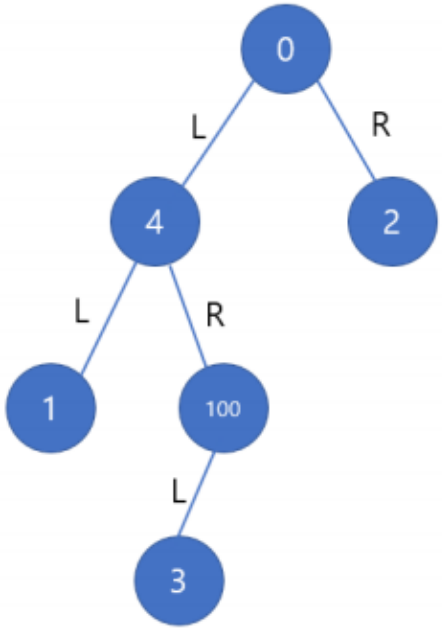
add RL 8 😊 1

add R 8 😊 -1

add LLLL 8 😊 -1

delete <pos_string>

동작 <pos_string> 위치에 존재하는 노드를 삭제한다. <pos_string>는 'L' 또는 'R'로 이루어진 길이 1 이상의 20 이하의 문자열로, add 함수에서와 같은 의미이다. 이 명령어는 항상 **단말 노드**만 삭제할 수 있다. 만약 <pos_string>에 존재하는 노드가 단말 노드가 아니라면 삭제에 실패하여 -1 을 출력한다. 가령, 트리의 상태가 아래와 같다면,



"delete LL"을 수행하면 데이터 1 을 가지는 노드가 삭제된다.

"delete R"을 수행하면 데이터 2 를 가지는 노드가 삭제된다.

"delete L"은 유효하지 않은 명령어이다. 데이터 4 를 가지는 노드는 단말 노드가 아니기 때문에 삭제할 수 없다. 단, "delete LL", "delete LRL", "delete LR"을 수행하여 자손 노드를 모두 삭제했다면 4 를 가지는 노드도 삭제할 수 있다.

"delete RR"은 유효하지 않은 명령어이다. 현재 트리에서 RR 위치에 (데이터 2 를 가지는 노드의 오른쪽 자식 노드) 노드가 존재하지 않기 때문이다.

delete 명세

출력 삭제에 성공하면 1 을 출력하고, <pos_string> 이 존재하지 않는 노드를 가리키거나, 내부 노드를 가리킬 경우 -1 을 출력한다.

예제 (현재 트리의 상태가 위 예와 같다면)

delete LL 😊 1

delete R 😊 1

delete L 😊 -1

delete RR 😊 -1

preorder

동작 현재 트리를 전위 순회하면서 노드에 저장된 데이터를 출력한다.

출력 전위 순회를 수행하면서 노드에 저장된 데이터를 띄어쓰기로 구분하여 출력한다. 구현의 편의를 위해 마지막 데이터를 출력하고 그 다음에 띄어쓰기를 출력하는 것을 허용한다.

예제 preorder 😊 0 4 1 100 3 2

inorder

동작 현재 트리를 중위 순회하면서 노드에 저장된 데이터를 출력한다.

출력 중위 순회를 수행하면서 노드에 저장된 데이터를 띄어쓰기로 구분하여 출력한다. 구현의 편의를 위해 마지막 데이터를 출력하고 그 다음에 띄어쓰기를 출력하는 것을 허용한다.

예제 inorder 😊 1 4 3 100 0 2

postorder

동작 현재 트리를 후위 순회하면서 노드에 저장된 데이터를 출력한다.

출력 후위 순회를 수행하면서 노드에 저장된 데이터를 띄어쓰기로 구분하여 출력한다. 구현의 편의를 위해 마지막 데이터를 출력하고 그 다음에 띄어쓰기를 출력하는 것을 허용한다.

예제 postorder 😊 1 3 100 4 2 0

exit

동작 프로그램을 종료한다.

출력 이 명령어의 출력은 없다.

예제 exit (프로그램 종료)

제출

- 기한: 2020 년 6 월 3 일 수요일 23:59
- 방법: 포털의 과제 제출란에 정해진 이름으로 압축 파일을 올린다. 학번과 이름이 20171001 김덕성이라면 HW2_20171010_김덕성.zip 으로 아래 파일을 압축하여 제출한다.
- **main.c**: 위의 명세를 구현한 소스 코드 파일
- **report.pdf**: 구현 방법을 요약한 보고서. 단, 보고서는 A4 용지로 2 장 이내로 제한한다.
- 딜레이는 전체 점수에서 1 일 이내(6 월 4 일 23:59)인 경우 20%, 3 일 이내(6 월 6 일 23:59)인 경우 50%를 감점한다.

Pair Programming

- 두 명에서 프로그래밍을 하되, 한 명은 타이핑을 한 명은 실시간 리뷰를 하는 개발 방법

- 지식 공유 & 피어 리뷰 (peer review)

프로그래밍 연습

- 실습용으로 BAEKJOON Online Judge를 사용하려고 합니다.
  - https://www.acmicpc.net/
  - 채점을 위해 미리 회원가입 필수

BAE/KJOON
ONLINE JUDGE

- 알고리즘 유형별 문제
  - https://www.acmicpc.net/problem/tags

태그	문제
다이나믹 프로그래밍	596
수학	349
구현	221
그래프 이론	176
그리디 알고리즘	173
BFS	137
브루트 포스	134
기하 알고리즘	132
시뮬레이션	127
정렬	113
이분 탐색	113
탐색	112
문자열 처리	108
DFS	102
다익스트라 알고리즘	80
자료구조	79
백트래킹	72
조합론	68
네트워크 플로우	62
플로이드 와샬 알고리즘	55
최대 유량	51
트리	43
입문용	43

연습 문제

- 문제를 같이 이해하고 풀어봅시다.

- 1662 압축
  - https://www.acmicpc.net/problem/1662

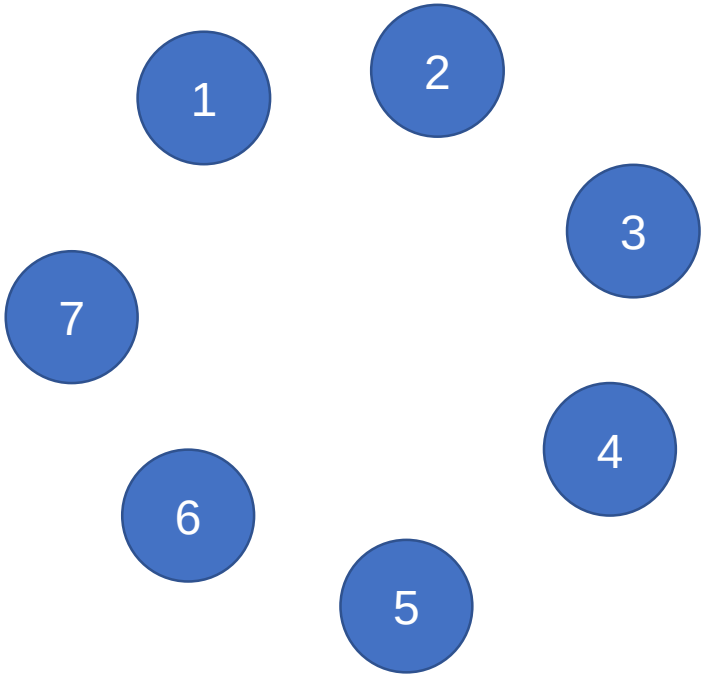
- 1158 요세푸스 문제
  - https://www.acmicpc.net/problem/1158

- 3020 개똥벌레
  - https://www.acmicpc.net/problem/3020

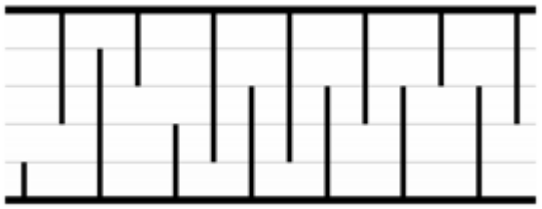
압축

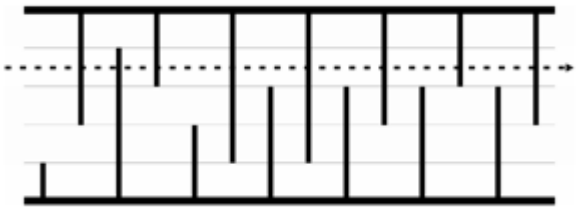
0	1	2	3	4	5	6	7	8	9	10	11	12	13
3	3	(	5	6	2	(	7	1	(	9	)	)	)

요세푸스 문제



개똥벌레





n = 7, h = 5

d = [1, 4, 2, 3, 3, 3, 3]

u = [3, 2, 4, 4, 3, 2, 3]