*알고리즘*

환자 관리 프로그램 설계

*컴퓨터공학과 12161558 김혜윤*

*cqqwer@gmail.com*

1. 개요
2. 설계의 목적
3. 2021-1 알고리즘 과목 설계 과제는 인하대병원의 환자 정보 관리 프로그램 설계로 레드블랙트리를 생성하고 관리하는 것이다.
4. 요구 사항

* 신규 가입 : 환자 번호, 이름, 연락처, 주소의 x좌표 / y좌표, 병명 그리고 진료비를 입력 받아서 동일한 환자 번호가 존재하면 그 노드의 깊이와 거절을 표시하는 0을 출력한다. 존재하지 않으면 추가하고 깊이와 승인을 표시하는 1을 출력한다.
* 환자 검색 : 환자 번호를 입력 받아서 존재하면 노드의 깊이와 이름, 연락처 그리고 주소를 출력하고, 존재하지 않으면 “Not found”를 출력한다.
* 추가 진료 : 환자 번호, 병명 그리고 진료비를 입력 받아서 동일한 환자 번호를 가진 노드가 존재하면 환자 정보에 병명과 진료비를 추가하고, 존재하지 않으면 “Not found”를 출력한다.
* 유행병 조사 : 입력 받은 병명과 마지막으로 추가된 병명과 일치한 환자의 수를 출력한다.

1. 개발 환경

* 운영체제 : MacOS Big Sur (v. 11.3.1)
* 개발언어 : C++
* 사용 프로그램 : Visual Studio Code

1. 필요한 자료구조 및 기능
2. 필요한 자료구조

* 환자 주소 좌표 저장 구조체
* //환자의 주소 좌표 저장 구조체
* struct addr
* {
* int x; // 주소의 x좌표 ( 0 ~10,000 사이의 임의의 정수)
* int y; // 주소의 y좌표 ( 0 ~10,000 사이의 임의의 정수)
* };
* 환자의 진료기록 저장 구조체
* //환자의 진료기록 저장 구조체
* struct history
* {
* string dName; // 병명 : 공백 없는 20 bytes 이내의 문자열
* int price; // 진료비 : 10,000 ~ 100,000 사이의 정수
* };
* 환자 정보 클래스
* //환자 정보 저장
* class Patient
* {
* public:
* int Pnum; //환자번호 (기준키, 유일함) : 1,000,000 ~ 1,999,999 사이의 정수
* string name; //이름 : 공백 없는 20bytes 이내의 문자열
* string phoneNum; //연락처 : 공백없는 11bytes 이내의 문자열
* addr Pad; //주소좌표
* history Phis; //진료기록
* };
* Linked-List를 위한 node 클래스
* //환자 정보를 가진 node
* class Node
* {
* public:
* Patient \*p; //환자 정보
* Node \*left; //왼쪽 자식 노드
* Node \*right; //오른쪽 자식 노드
* Node \*parent; //부모 노드
* //각 노드의 색상표시를 위해서 (0: Black, 1:Red)
* int color; //색상 변수
* };

1. 기능

* 신규 환자 추가
  1. 입력 : I K N H Ax Ay DI C (신규가입 기호, 환자 번호, 이름, 연락처, 주소의 x좌표,주소의 y좌표, 병명, 진료비)
  2. root 노드부터 시작해서 환자 번호 비교를 통해 왼쪽 서브 트리, 오른쪽 서브 트리로 진행하면서 마지막 노드를 찾아 자식 노드로 추가한 후, 부모 노드의 color 변수가 0(black)일 때까지 삼촌 노드의 색상에 따라 recoloring 또는 restructing을 진행한다
  3. 출력 : 환자 정보의 노드의 깊이, 처리를 나타내는 정수 (0: 거절, 1: 승인)
* 환자 검색
  + 1. 입력 : F K (검색 기호, 환자 번호
    2. 입력 : F K (검색 기호, 환자 번호)
    3. Root 노드부터 시작해서 leaf 노드가 나올 때까지 환자 번호 비교로 탐색한다. 내려갈 때마다 depth++를 해줌으로써 깊이를 찾는다.
    4. 출력 : D N H Ax Ay (노드의 깊이, 이름, 연락처, 주소의 x좌표, 주소의 y좌표) 또는 not found
* 특정 환자의 새로운 진료 정보 추가

(1) 입력 : A K DI C (기호, 환자 번호, 병명, 진료비) , 입력 받은 병명을 탐색한 환자의 진료 기록에 추가

(2) Root 노드부터 시작해서 leaf 노드가 나올 때까지 환자 번호 비교로 탐색한다. 내려갈 때마다 depth++를 해줌으로 깊이를 찾고, 일치하는 노드를 발견하면 노드의 환자 정보의 병명과 진료비를 추가해준다.

(3) 출력 : D (노드의 깊이) 또는 not found

* 유행병 조사

(1) 마지막에 진료받은 병명을 기준으로 특정 병명의 수를 집계

(2) 입력 : E DI (기호, 병명)

(3) Root 노드부터 시작해서 leaf 노드가 나올 때까지 재귀 호출을 통해서 왼쪽 서브 트리와 오른쪽 서브 트리를 순회한다. 각 노드의 병명을 비교해서 일치하면 전역 변수 dcnt의 값을 증가시켜준다;

* + 1. 출력 : T (환자의 수 >=0)

1. 기능별 알고리즘 명세
2. 기능을 위한 알고리즘 설명 (복잡도 분석)
3. 알고리즘 설명

* 이진 탐색 알고리즘
  + - * 이진 탐색이란 데이터가 정렬 되어있는 배열에서 특정한 값을 찾아내는 알고리즘이다. 배열의 중간에 있는 임의의 값을 선택하여 찾고자 하는 값과 배교한다. 찾고자 하는 값보다 작으면 기준으로 왼쪽 데이터들을 대상으로, 크면 오른쪽 데이터들을 대상으로 다시 탐색한다. 해당 값을 찾을 때까지 이 과정을 반복한다.

1. 기능별 시간 복잡도
   1. 신규가입

Insert O(log n)

FixUp (Left / Right Rotation) O(1)

* + - * + O(log n) 이다.
  1. 환자 검색, 추가 진료

이진 탐색 알고리즘을 이용하기 때문에

O(log n) 이다.

* 1. 유행병 조사

중위 순회를 이용하기 때문에

최악의 경우 O(N) 이다.

1. 인터페이스 및 사용법
2. 간단한 사용법 설명

* 수행할 횟수 입력한다.
* 수행 할 기능 선택한다. (“I” : 신규 가입, “F” : 환자 탐색, “A” : 추가 진료, “E” : 유행병 조사)
* 선택한 기능에 따라 필요한 정보들을 입력한다. (“I” : 환자 번호, 이름, 전화번호, 주소 좌표, 병명, 진료비, “F” : 환자 번호, “A” : 환자 번호, 병명, 진료비, “E” : 병명)

1. 실행 화면 캡쳐 포함

* 텍스트이(가) 표시된 사진

  자동 생성된 설명![테이블이(가) 표시된 사진

  자동 생성된 설명]()

1. 평가 및 개선 방향
2. 구현한 알고리즘의 장점

* Linked-List 기반으로 구현하였음으로 메모리 사용량이 적고 실행 시간이 빠르다.
* Leaf 노드를 만났을 때, 탐색을 멈추므로, 빠른 수행 시간을 보여줄 것으로 예상한다.

1. 구현한 알고리즘의 단점

* 노드를 탐색하고 결과로 노드의 깊이를 출력하기 위해서 같은 코드들이 다른 함수 내에 중복으로 존재한다.

1. 향후 개선 방향

* 노드 탐색과 깊이 탐색을 독자적인 함수로 만들고 필요한 부분에서 호출하도록 할 수 있다면 코드가 훨씬 단순하고 효율적일 것이라고 예상된다.
* 새로운 노드를 추가하고 fixup을 수행할 때, If / else 문을 줄일 수 있다면 효율적일 것으로 예상된다.