2023 Algorithm Study 7Week

sookyoung0620 · 방금 전 · 비공개

2023 Algorithm Study

목록 보기 7/7

탐색트리

- 이진탐색트리 : 이진트리란 최상위 루트 노드(트리 구조에서 데이터가 들어가는 한 칸)로부터 하위 트리로 갈 때 가질 수 있는 자식의 노드수가 2개 이하인 트리구조. 즉 왼쪽, 오른쪽 아래쪽으로 자식 노드를 가질 수 있는 구조
- 적정이진트리(리프 노드를 제외한 모든 노드들이 0개 혹은 2개의 자식 노드를 가진 트리)로 구현
- 중회순회하면 키가 증가하는 순서로 방문
- 탐색
  - 루트에서 출발하는 하향 경로
  - 키 k와 현재 노드의 키의 크기를 비교하여 다음 방문할 노드 결정
    - k보다 크면 작으면 왼쪽 자식 탐색, k보다 작으면 오른쪽 자식 탐색
  - 외부모드에 다다르면 키 k는 발견되지 않은 것
- 삽입
  - k가 트리에 존재하지 않을 경우, 탐색은 외부노드에 도착
  - 외부노드 w에 k를 삽입한 후 w를 내부노드로 확장
- 삭제
  - k가 트리에 존재할 경우, 키가 있던 노드의 빈 자식 노드를 지우고 k가 있던 노드도 삭제
  - k가 내부노드만을 자식들로 가지는 노드에 존재할 경우, k가 있는 노드의 오른쪽 부트리 내 노드 중 가장 왼쪽으로 돌출된 내부노드의 내용을 k가 있는 노드에 복사

1620

- 문제 접근

1. 탐색이진트리 : 왼쪽, 오른쪽 자식 노드 사용
2. 인덱스와 이름 : 이름과 인덱스의 연결성

- 문제 풀면서

1. 노드를 만들었는데 인덱스+1하면 바로 끝날 줄 알았는데 생각해보니 배열이 아니라 따로 저장해야함... 포인터로 받는 것도 도전하였으나 자꾸 문자를 입력했을 때 숫자가 출력되는 것이 안됨

```
 > #include <stdio.h>
#include <string.h>
#include <stdlib.h>
#pragma warning(disable: 4996)

typedef struct Node {
    int key;
    char name[21];
    struct Node* lChild;
    struct Node* rChild;
} Node;

Node* tree;

Node* treeSearch(Node* n, int k) {
    if (n == NULL)
        return NULL;

    if (k == n->key) {
        return n;
    }
    else if (k < n->key) {
        if (treeSearch(n->lChild, k) == NULL) {
            return n;
        }
        else {
            return treeSearch(n->lChild, k);
        }
    }
    else {
        if (treeSearch(n->rChild, k) == NULL) {
            return n;
        }
        else {
            return treeSearch(n->rChild, k);
        }
    }
}

Node* nameSearch(Node* root, char* name) {
    if (root == NULL) {
        return NULL;
    }

    int cmp = strcmp(name, root->name);

    if (cmp == 0) {
        return root;
    }
    else if (cmp < 0) {
```

```
        return nameSearch(root->lChild, name);
    }
    else {
        return nameSearch(root->rChild, name);
    }
}

int main() {
    int n, m;
    char input[21];

    scanf("%d %d", &n, &m);

    tree = NULL;

    for (int i = 1; i <= n; i++) {
        Node* newNode = (Node*)malloc(sizeof(Node));
        newNode->key = i;
        scanf("%s", newNode->name);
        newNode->lChild = NULL;
        newNode->rChild = NULL;

        if (tree == NULL) {
            tree = newNode;
        }
        else {
            Node* w = treeSearch(tree, i);

            if (i < w->key)
                w->lChild = newNode;
            else
                w->rChild = newNode;
        }
    }

    for (int i = 0; i < m; i++) {
        scanf("%s", input);

        if (input[0] >= '0' && input[0] <= '9') {
            int id = atoi(input);
            Node* result = treeSearch(tree, id);
            if (result != NULL) {
                printf("%s\n", result->name);
            }
            else {
                printf("Not Found\n");
            }
        }
        else {
            Node* result = nameSearch(tree, input);
            if (result != NULL) {
                printf("%d\n", result->key);
            }
            else {
                printf("Not Found\n");
            }
        }
    }

    return 0;
}
```

2. 입출력에서 포켓몬이 출력이 안됨

```
 > #pragma warning (disable:4996)
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct Node {
    int key;
    char name[21];
    struct Node* lChild;
    struct Node* rChild;
} Node;

Node* tree;

Node* treeInsert(Node* root, int k, char* name) {
    if (root == NULL) {
        Node* newNode = (Node*)malloc(sizeof(Node));
        newNode->key = k;
        strcpy(newNode->name, name);
        newNode->lChild = NULL;
        newNode->rChild = NULL;
        return newNode;
    }

    if (strcmp(name, root->name) < 0) {
        root->lChild = treeInsert(root->lChild, k, name);
    }
    else {
        root->rChild = treeInsert(root->rChild, k, name);
    }

    return root;
}

Node* treeSearch(Node* n, int k) {
    if (n == NULL)
        return NULL;

    if (k == n->key) {
        return n;
    }
    else if (k < n->key) {
        return treeSearch(n->lChild, k);
    }
    else {
        return treeSearch(n->rChild, k);
    }
}

Node* nameSearch(Node* root, char* name) {
    if (root == NULL) {
        return NULL;
    }

    int cmp = strcmp(name, root->name);

    if (cmp == 0) {
        return root;
    }
```

```
        else if (cmp < 0) {
            return nameSearch(root->lChild, name);
        }
        else {
            return nameSearch(root->rChild, name);
        }
    }

    int main() {
        int n, m;
        char input[21];

        scanf("%d %d", &n, &m);

        tree = NULL;

        for (int i = 1; i <= n; i++) {
            char name[21];
            scanf("%s", name);
            tree = treeInsert(tree, i, name);
        }

        for (int i = 0; i < m; i++) {
            scanf("%s", input);

            if (input[0] >= '0' && input[0] <= '9') {
                int id = atoi(input);
                    Node* result = treeSearch(tree, id);
                    if (result != NULL) {
                        printf("%s\n", result->name);
                    }

            }
            else {
                Node* result = nameSearch(tree, input);
                if (result != NULL) {
                    printf("%d\n", result->key);
                }

            }
        }


    return 0;
}
```

윤수경

2023 Algorithm Study 5Week

0개의 댓글

댓글을 작성하세요

댓글 작성