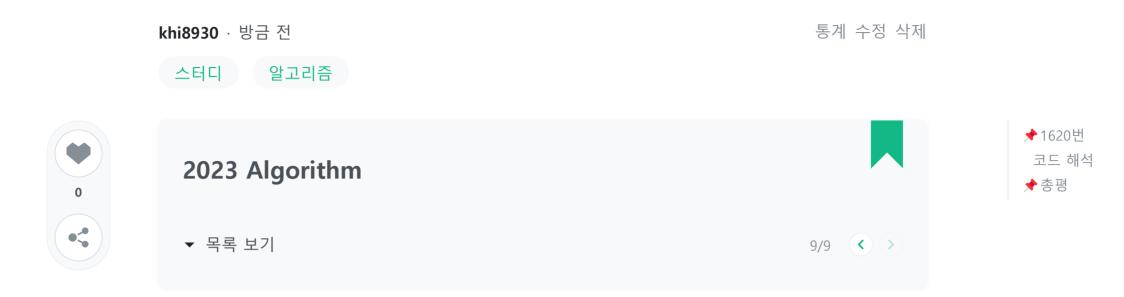
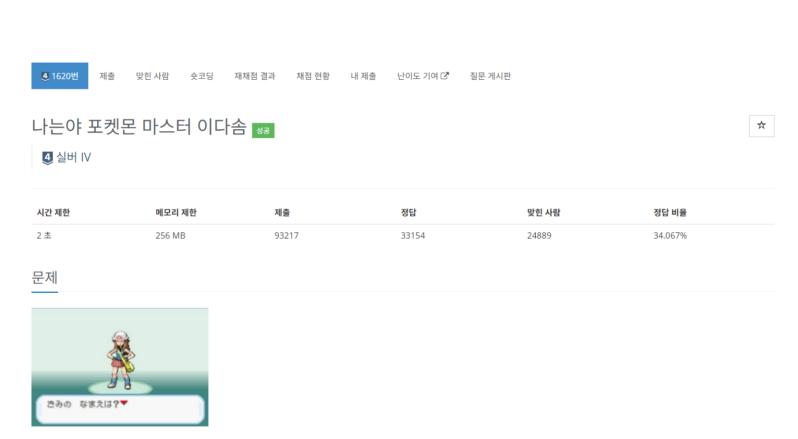
백준 | 1620번 나는야 포켓몬 마스터 이다솜



- 소요시간 : 2시간
- 자료 참고 여부 : △ (자료참고는 했으나, 직접적으로 사용 X)
- 체감 난이도 : 中





- https://www.acmicpc.net/problem/1620
- 1. 문제는 풀었지만 원하는데로 풀지는 못했음..
- 2. 왜 더 어렵게 풀면 시간 초과인거니 .. ?

코드 전문

```
#pragma warning(disable:4996)
#pragma warning(disable:4013)
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
typedef struct Node {
    char key[21];
    int num;
} node;
int compare(const void* a, const void* b) {
    return strcmp(((node*)a)->key, ((node*)b)->key);
int main() {
    node* root = NULL;
    node* nSorted = NULL;
    char k[21] = \{ 0 \};
    char p[21] = \{ 0 \};
    int n, m;
    node* r;
    scanf("%d %d", &n, &m);
    root = (node*)calloc(n+1, sizeof(node));
    nSorted = (node*)calloc(n + 1, sizeof(node));
    for (int i = 0; i < n; i++) {
```

```
scanf("%s", root[i].key);
    root[i].num = i+1;
    nSorted[i] = root[i];
qsort(root, n, sizeof(node), compare);
// printf("----\n");
for (int i = 0; i < m; i++) {
    scanf("%s", p);
    if (atoi(p) > 0) {
        int swit = atoi(p);
        printf("%s\n", nSorted[swit-1].key);
    }
    else {
        int left, right, mid;
        left = 0, right = n - 1;
        while (left <= right) {</pre>
            mid = (left + right) / 2;
            int compare_result = strcmp(root[mid].key, p);
            if (compare_result == 0) {
                 printf("%d\n", root[mid].num);
                 break;
            }
            else if (compare_result > 0) {
                 right = mid - 1;
            }
            else {
                left = mid + 1;
    }
return 0;
```

코드 해석

```
#pragma warning(disable:4996)
#pragma warning(disable:4013)
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
typedef struct Node {
   char key[21];
   int num;
} node;
int compare(const void* a, const void* b) {
   return strcmp(((node*)a)->key, ((node*)b)->key);
int main() {
   node* root = NULL;
   node* nSorted = NULL;
   char k[21] = \{ 0 \};
   char p[21] = \{ 0 \};
   int n, m;
   node* r;
   scanf("%d %d", &n, &m);
   root = (node*)calloc(n+1, sizeof(node));
   nSorted = (node*)calloc(n + 1, sizeof(node));
   for (int i = 0; i < n; i++) {
        scanf("%s", root[i].key);
        root[i].num = i+1;
        nSorted[i] = root[i];
```

- void compare
 - node *a와 node* b의 크기를 비교하기 위한 함수
 - 두 노드 key(문자열)의 비교 결과에 따라 0이 나오거나 0보다 큰 값이 나오거나, 0보다 작은 값이 나온다.
- main 함수
 - n,m을 입력 받은 뒤 그 크기에 맞게 두 개의 포인터 배열을 동적할당 한다.
 - root 배열에 먼저 입력을 받은 뒤, 그 값을 nSorted 함수로 동일하게 옮겨준다.
 - 이후 root 함수는 정렬되고, nSorted 함수는 정렬되지 않는다.

```
qsort(root, n, sizeof(node), compare);

for (int i = 0; i < m; i++) {
    scanf("%s", p);
    if (atoi(p) > 0) {
        int swit = atoi(p);
    }
}
```

```
printf("%s\n", nSorted[swit-1].key);
    }
    else {
        int left, right, mid;
        left = 0, right = n - 1;
        while (left <= right) {</pre>
            mid = (left + right) / 2;
            int compare_result = strcmp(root[mid].key, p);
            if (compare_result == 0) {
                printf("%d\n", root[mid].num);
                break;
            else if (compare_result > 0) {
                right = mid - 1;
            else {
                left = mid + 1;
return 0;
```

- qsort 함수 (https://twpower.github.io/56-qsort-in-c)
 - void gsort (void base, size_t nel, size_t width, int (compare)(const void , const void);
 - 위와 같은 기본 구조를 가지며, 정렬해주는 함수이다.
 - qsort(root, n, sizeof(node), compare); → n개의 요소를 가진 root 노드를 compare에 따라서 정렬시킨다.
- char형 p입력
 - char 형의 p를 입력 받은 후 int 형으로 변환해본다.
 - 만약 이 결과가 자연수 값이면 숫자를 입력한 것이기 때문에 숫자에 맞는 포켓몬을 출력한다.
 - 자연수가 나오지 않는다면, 이는 문자열을 입력한 것이기 때문에 특정 문자열에 맞는 숫자를 출력한다.
- 문자열 입력시 해당 숫자 출력하기 (else 문)
 - 이진 탐색을 사용한다. middle 값을 정하고 입력 받은 문자열과 root의 특정 문자열을 비교하여 이진탐색을 진행한다.

(실패) 탐색트리를 사용한 코드

```
#pragma warning(disable:4996)
#pragma warning(disable:4013)
#include<stdio.h>
#include<string.h>
#include <stdlib.h>
typedef struct Node {
   struct Node* left;
   struct Node* right;
   struct Node* parent;
   char key[21];
   int num;
}node;
node* root;
int CNT = 0;
void init() { // 이중 포인터
    root = (node*)malloc(sizeof(node));
    root->right = NULL;
   root->left = NULL;
    root->parent = NULL;
int isExternal(node* w) {
   if (w->left == NULL && w->right == NULL) {
        return 1;
   }
   else {
        return 0;
}
int isInternal(node* w) {
   if (w->left != NULL || w->right != NULL) {
        return 1;
   else {
        return 0;
```

```
node* treeSearch(node* v, int k) { // 별 찍는거
   if (isExternal(v)) {
        return v;
   if (k == (v->num)) {
       return v;
   else if (k < (v->num)) {
       return treeSearch(v->left, k);
   else {
       return treeSearch(v->right, k);
void insertItem(node* root, char *k, int no) {
   node* n_left = (node*)malloc(sizeof(node));
   node* n_right = (node*)malloc(sizeof(node));
   node* w = treeSearch(root, no);
   if (isInternal(w)) {
        return;
   else {
       strcpy(w->key, k);
       w->num = no;
       w->left = n_left;
       w->right = n_right;
        n_left->parent = w;
        n_right->parent = w;
        n_left->left = NULL;
        n_left->right = NULL;
       n_right->left = NULL;
       n_right->right = NULL;
void printTree_arr(node* p, char *k) {
   if (isExternal(p)) {
        return;
   else {
       if (strcmp(k, p->key) == 0) {
           printf("%d", p->num);
       }
        printTree_arr(p->left, k);
        printTree_arr(p->right, k);
}
void printTree_num(node* p, int k) {
   if (isExternal(p)) {
        return;
   else {
       if (k == p->num) {
           printf("%s", p->key);
       printTree_num(p->left, k);
        printTree_num(p->right, k);
}
int main() {
   char p[21] = { 0 };
   int n, m;
   char k[21] = \{ 0 \};
   root = (node*)malloc(sizeof(node));
   root->parent = NULL;
   root->left = NULL;
   root->right = NULL;
   scanf("%d %d", &n, &m);
   for (int i = 0; i < n; i++) {
       scanf("%s", k);
       insertItem(root, k, i+1);
        getchar();
   //printf("----\n");
```

```
for (int i = 0; i < m; i++) {
    scanf("%s", p);
    int s_number = atoi(p);
    getchar();

    if (s_number > 0) {
        printTree_num(root, s_number);
    }
    else {
        printTree_arr(root, p);
    }
    printf("\n");
}
```

총평

- 수업시간에 사용한 탐색트리 ADT를 사용해서도 구현해보고, 동적할당과 비재귀 방식으로 바꿔서 실행시켜도 봤는데 계속 시간 초과가 나왔다.
- 계속 시도하다 실패해서, 한 블로그를 참고해 풀이할 때 사용한 자료구조만을 사용해서 이진트리로 풀었는데 탐색트리로 풀지 못해 아쉽다.



향임 코딩 공부 중인 대학생 🔔



0개의 댓글

댓글을 작성하세요

댓글 작성

