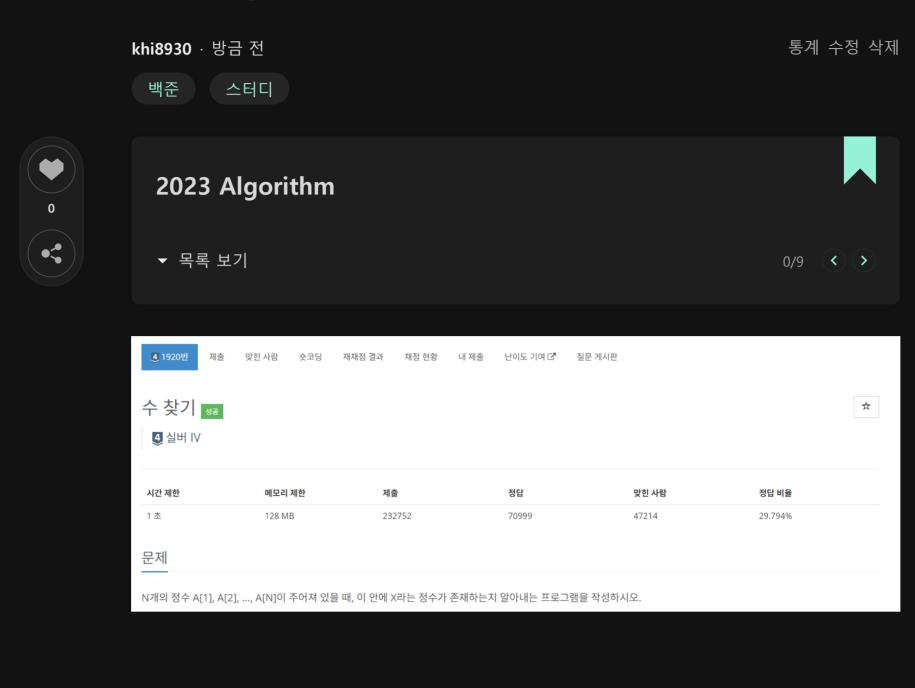


★1620번
코드 해석

📌 총평



백준 | 1920번 수 찾기



- 소요시간 : 1시간
- 자료 참고 여부 : △ (자료참고는 했으나, 직접적으로 사용 X)
- 체감 난이도 : 中

★ 1620번



• https://www.acmicpc.net/problem/1920

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
69154248	khi8930	4 1920	맞았습니다!!	4312 KB	92 ms	C99 / 수정	1538 B	14분 전
69154150	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C99 / 수정	1509 B	19분 전
69153563	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C99 / 수정	1329 B	42분 전
69153405	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	1465 B	47분 전
69153353	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	1326 B	49분 전
69152816	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	1385 B	1시간 전
69152245	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	1473 B	1시간 전
69152157	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	1378 B	1시간 전
69152145	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	1386 B	1시간 전
69152112	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C99 / 수정	1367 B	1시간 전
69151978	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	1367 B	1시간 전
69151925	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	1354 B	1시간 전
69151861	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	1626 B	1시간 전
69151796	khi8930	4 1920	<u>런타임 에러 (Segfault)</u>			C++17 / 수정	2456 B	1시간 전

- 1. 지긋지긋한 런타임 에러의 늪 ..
- 2. 또또또 기억력에 의존해서 풀려고 하다 .. ㅜ_ㅜ

```
#pragma warning(disable:4996)
#pragma warning(disable:4013)
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#define SIZE 10001
typedef struct NODE {
   int data;
   struct NODE* NEXT;
} node;
int M;
node* HashTable;
int hashFunction(int key) {
   if (key < 0) {
       key *= -1;
   return key % SIZE;
void insertItem(int k) {
   int v = hashFunction(k);
   node* n, * p;
   n = (node*)malloc(sizeof(node));
   n->data = k;
   n->NEXT = NULL;
   p = HashTable + v;
   n->NEXT = p->NEXT;
   p->NEXT = n;
void FindItem(int k) {
   int f = 0;
   int v = hashFunction(k);
   node* p;
   p = HashTable + v;
   int cnt = 0;
   while (1) {
       if (p->NEXT == NULL) {
           cnt = 0;
       p = p->NEXT;
       cnt++;
       if (p->data == k) {
   if (cnt > 0) {
       printf("1\n");
       printf("0\n");
int main() {
   int k, N;
   scanf("%d", &M);
   HashTable = (node*)malloc(sizeof(node)*SIZE );
   for (int i = 0; i < SIZE; i++) {
       HashTable[i].data = 0;
       HashTable[i].NEXT = NULL;
   for (int i = 0; i < M; i++) {
       scanf("%d", &k);
       insertItem(k);
   scanf("%d", &N);
   for (int i = 0; i < N; i++) {
       scanf("%d", &k);
       FindItem(k);
   free(HashTable);
   return 0;
```

```
#define SIZE 10001

typedef struct NODE {
    int data;
    struct NODE* NEXT;
} node;

int M;
node* HashTable;

int hashFunction(int key) {
    if (key < 0) {
        key *= -1;
    }
    return key % SIZE;
}</pre>
```

'알고리즘' 수업에서 했던 Hash Function의 사용과 이 문제에서 사용하는 방법이 달라서 런타임에러가 계속 떴다.. 해시 테이블에 대한 이해 부족으로 일어난 일..

• Hash Function을 이용해서 Hash 함수 값을 만들어 준다. 입력받은 key 값에 최대 Size를 나누는 것을 Hash Function으로 사용한다.

```
void insertItem(int k) {
   int v = hashFunction(k);
   node* n, * p;
   n = (node*)malloc(sizeof(node));
   n->data = k;
   n->NEXT = NULL;
   p = HashTable + v;
   n->NEXT = p->NEXT;
   p->NEXT = n;
void FindItem(int k) {
   int f = 0;
   int v = hashFunction(k);
   node* p;
   p = HashTable + v;
   int cnt = 0;
   while (1) {
       if (p->NEXT == NULL) {
           cnt = 0;
       p = p->NEXT;
       cnt++;
        if (p->data == k) {
   if (cnt > 0) {
        printf("1\n");
   else {
        printf("0\n");
```

• 해시 테이블에서 사용한 InsertItem과 FindItem은 '알고리즘' 수업에 사용한 것 그대로 사용했다. 다만, 두 함수 모두 Hash Function을 사용해서 해시 값을 정한다는 점과 FindItem에서 깊이를 출력하는 것이 아니라 값의 존재에 따라 숫자를 출력 한다는 점이 차이점이다.

```
int main() {
    int k, N;

    scanf("%d", 6M);

HashTable = (node*)malloc(sizeof(node)*SIZE);

for (int i = 0; i < SIZE; i++) {
        HashTable[i].data = 0;
        HashTable[i].NEXT = NULL;
}

for (int i = 0; i < M; i++) {
        scanf("%d", 6k);
        insertItem(k);
}

scanf("%d", 6N);

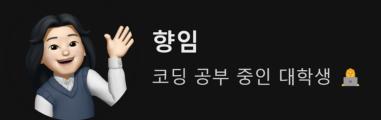
for (int i = 0; i < N; i++) {
        scanf("%d", 6k);
        FindItem(k);
}</pre>
```

free(HashTable);

- 입력 받은 Size(M) 따라서 Hash Table을 동적할당 뒤, M개의 변수를 입력받는다.
- N을 입력받아서 N번의 FindItem을 실행시킨다.

ᅔᇂ평

- 수업에서 사용한 해시 테이블 알고리즘을 그대로 사용하면 되는 문제여서 풀이가 쉽게 진행됬다.
- 해시 함수를 알고리즘 문제에서 풀었던 것과 똑같이 써서 계속 런타임에러가 나왔었는데, 이 부분은 해시 테이블에 대한 이해가 부족해서 그랬다 ㅜ_ㅜ.. 공부를 더할 필요가 있다 ..





이전 포스트

백준 | 1620번 나는야 포켓몬 마...

0개의 댓글

댓글을 작성하세요

댓글 작성

