# 2023 Algorithm Study 8Week



sookyoung0620 · 방금 전 · 비공개





## 해시테이블

- 키-주소 매핑에 의해 구현된 사전 ADT
- 버켓 배열 + 해시함수
- 탐색, 삽입, 삭제 : 최악시간 O(n), 기대시간 O(1)
- 버켓 배열
  - 크기 M의 배열 A
  - M == 배열의 용량
  - 키 k를 가진 원소 e는 버켓 A[k]에 삽입
- 해시함수
  - 정수 h(x) : 키 x의 해시값
  - 해시코드맵 h1 : keys->integers
  - 압축맵 h2 : integers->[0,M-1]
  - h(k) = h2(h1(k)) : 해시코드맵 -> 압축맵
  - 예 : 학번 -> 마지막 수 4자리 수 -> 방번호
  - 좋은 해시 함수 : 키들을 외견상 무작위하게 분산, 계산이 빠르고 쉬워야함

## 백준 1920번 : 수 찾기

1. 맨 처음에 N,M을 전역변수로 선언했는데 출력이 안 나와서(아예 아무것도 안 나옴) N을 함수에 다 추가해봄

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



typedef struct node {
    int key;
    struct node* next;
}Node;

Node* hashTable;

void init(int N) {
    for (int i = 0; i < N; i++) {
        (hashTable + i)->next = NULL;
    }
}

int h(int k, int N) {
    return k % N;
}

void insertItem(int x) { //탐색
    int v = h(x,N);

    Node* p = hashTable + v;
    Node* newnode = (Node*)malloc(sizeof(Node));
    newnode->key = x, newnode->next = NULL;

    if (p->next == NULL) {
        p->next = newnode;
    }
    else {
        newnode->next = p->next;
        p->next = newnode;

    }
}

int searchItem(int x) { //삽입
    int v = h(x,N);
    int cnt = 0;
    Node* p = hashTable + v;
    if (p->next == NULL) {
        return 0;
    }
    else {
        while (1) {
            p = p->next;
            cnt++;
            if (p->key == x) {
                return 1;
            }
            if (p->next == NULL) {
                return 0;
```

```
                }
            }
        }
    }



int main() {

    int N, M;

    scanf("%d", &N);
    hashTable = (Node*)malloc(sizeof(Node) * N);
    init(N);


    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        insertItem(key);
    }
    scanf("%d", &M);
    for (int i = 0; i < M; i++) {
        int num;
        scanf("%d", &num);
        if (searchItem(num) == 1) {
            printf("1\n");
        }
        else {
            printf("0\n");
        }
    }


    return 0;
}
```

2. 하지만 여전히 출력이 아예 안 나오는 상황... 해시함수와 삽입, 탐색함수만으로 할 수 있다고 생각하기 때문에 검토함.. 문제 없는 거 같은데 괜히 N을 뺀게 신경쓰여서 N만 다시 전역변수로 뺌 (사실 동적 할당 전에 N을 입력 안받아서 오류도 있었음 하하)

3. 메인 함수에서 key, num을 반복문 안에서 선언한 것도 뭔가 찜찜해서 밖으로 뺐음
   4.왐마 출력예시대로 나옴.. 하지만 런타임 에러 그리고

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

int N;
typedef struct node {
    int key;
    struct node* next;
```

```
} Node;

Node* hashTable;

void init() {
    for (int i = 0; i < N; i++) {
        (hashTable + i)->next = NULL;
    }
}

int h(int k) {
    return k % N;
}

void insertItem(int x) { //삽입
    int v = h(x);

    Node* p = hashTable + v;
    Node* newnode = (Node*)malloc(sizeof(Node));
    newnode->key = x, newnode->next = NULL;

    if (p->next == NULL) {
        p->next = newnode;
    }
    else {
        newnode->next = p->next;
        p->next = newnode;
    }
}

int searchItem(int x) { //탐색
    int v = h(x);
    int cnt = 0;
    Node* p = hashTable + v;
    if (p->next == NULL) {
        return 0;
    }
    else {
        while (1) {
            p = p->next;
            cnt++;
            if (p->key == x) {
                return 1;
            }
            if (p->next == NULL) {
                return 0;
            }
        }
    }
}

int main() {
    int M;

    scanf("%d", &N);
    hashTable = (Node*)malloc(sizeof(Node) * N);
    init();
```

```
            int key;

            for (int i = 0; i < N; i++) {
                scanf("%d", &key);
                insertItem(key);
            }

            scanf("%d", &M);
            int num;
            for (int i = 0; i < M; i++) {
                scanf("%d", &num);
                if (searchItem(num) == 1) {
                    printf("1\n");
                }
                else {
                    printf("0\n");
                }
            }


    free(hashTable);

    return 0;
}
```

## 5. Table SIZE 정의해서 해시테이블을 만들어봄..

```
 > #define TABLE_SIZE 100000

typedef struct Node {
    int key;
    struct Node* next;
} Node;

Node* hashTable;

void init() {
    for (int i = 0; i < TABLE_SIZE; i++) {
        (hashTable + i)->next = NULL;
    }
}

int h(int k) {
    return abs(k) % TABLE_SIZE;
}

void insertItem(int x) {
    int v = h(x);

    Node* p = hashTable + v;
    Node* newnode = (Node*)malloc(sizeof(Node));
    newnode->key = x;
    newnode->next = p->next;
    p->next = newnode;
}
```

```
int searchItem(int x) {
    int v = h(x);
    Node* p = hashTable + v;

    while (p->next != NULL) {
        p = p->next;
        if (p->key == x) {
            return 1;
        }
    }

    return 0;
}

int main() {
    int N,M;

    scanf("%d", &N);
    hashTable = (Node*)malloc(sizeof(Node) * TABLE_SIZE);
    init();

    int key;

    for (int i = 0; i < N; i++) {
        scanf("%d", &key);
        insertItem(key);
    }

    scanf("%d", &M);
    int num;
    for (int i = 0; i < M; i++) {
        scanf("%d", &num);
        if (searchItem(num) == 1) {
            printf("1\n");
        } else {
            printf("0\n");
        }
    }


   free(hashTable);

   return 0;
}
```

6. 찾아보니까 입력된 정수 중 음수가 존재하는 경우 해시 값을 양수로 변환해야해서 abs 함수를 사용하라고 그랬다.. 성공

```
 >
#define TABLE_SIZE 100000

typedef struct Node {
    int key;
    struct Node* next;
} Node;
```

```
Node* hashTable;

void init() {
    for (int i = 0; i < TABLE_SIZE; i++) {
        (hashTable + i)->next = NULL;
    }
}

int h(int k) {
    return abs(k) % TABLE_SIZE;
}

void insertItem(int x) {
    int v = h(x);

    Node* p = hashTable + v;
    Node* newnode = (Node*)malloc(sizeof(Node));
    newnode->key = x;
    newnode->next = p->next;
    p->next = newnode;
}

int searchItem(int x) {
    int v = h(x);
    Node* p = hashTable + v;

    while (p->next != NULL) {
        p = p->next;
        if (p->key == x) {
            return 1;
        }
    }

    return 0;
}

int main() {
    int N,M;

    scanf("%d", &N);
    hashTable = (Node*)malloc(sizeof(Node) * TABLE_SIZE);
    init();

    int key;

    for (int i = 0; i < N; i++) {
        scanf("%d", &key);
        insertItem(key);
    }

    scanf("%d", &M);
    int num;
    for (int i = 0; i < M; i++) {
        scanf("%d", &num);
        if (searchItem(num) == 1) {
            printf("1\n");
        } else {
```

```
                printf("0\n");
            }
        }


    free(hashTable);

    return 0;
}
```

## 윤수경

이전 포스트
2023 Algorithm Study 7Week

## 0개의 댓글

댓글을 작성하세요

댓글 작성