자료구조

과제 8

물리학과 20182326 이선민

문제 1번 해싱. 결과

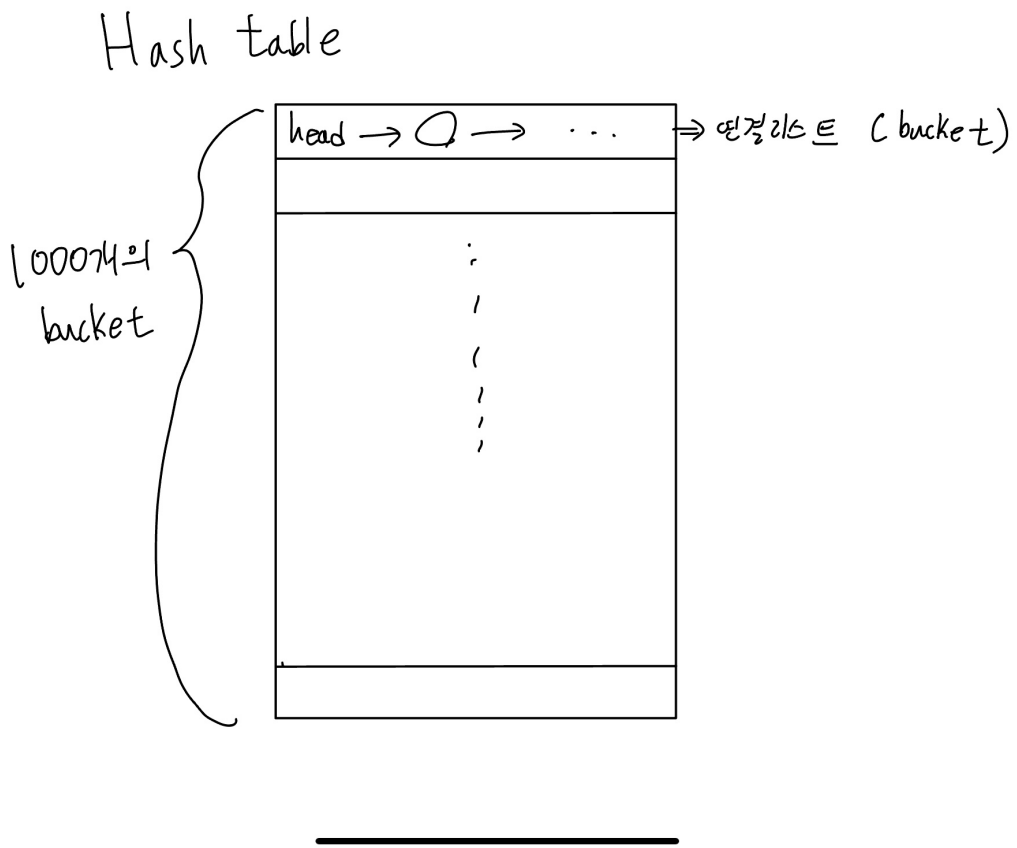
```
[~/2022_1/Data_structure/Assignment/assignment_8]$ gcc problem_1.c
[~/2022_1/Data_structure/Assignment/assignment_8]$ ./a.out     *[master]
해 싱  테 이 블 이  완 성 되 었 습 니 다 .
단 어 를  입 력 하 세 요  add
(2회  검 색 )  vt.추 가 하 다
단 어 를  입 력 하 세 요  hello
(22회  검 색 )  int.여 보
단 어 를  입 력 하 세 요  go
(1회  검 색 )  vi.가 다
단 어 를  입 력 하 세 요  unrobe
(0회  검 색 )  vt.-으  옷 을  벗 기 다  vi.옷 을  벗 다
단 어 를  입 력 하 세 요  temperate
(294회  검 색 )  a.지 나 치 지  않 는
```

구현 설명

```
 6 #define ARR_SIZE 500
 7
 8 struct _bucket* hashTable = NULL;
 9 int BUCKET_SIZE = 1000; // 버켓의 총 길이
10
11 typedef struct _node {
12     int key;
13     char eng[ARR_SIZE];
14     char kor[ARR_SIZE];
15     struct _node* next;
16 } Node;
17
18
19 typedef struct _bucket{
20     struct _node* head;
21     int count;
22 } Bucket;
23
```

- 우선 해시 테이블은 전역 변수로 선언해두었다. 그리고 테이블의 사이즈는 1000으로 하였다.
- 해시 테이블의 충돌을 연결리스트로 해결하였고 그러기 위해 node를 하나 선언 하여 그 안에 해시 테이블에 넣을 때 필요한 테이블 키값, 그리고 내용, 그리고 다음 노드를 가리키는 포인터를 선언해놓았다.
- Bucket구조체는 해시 테이블에서 각각의 연결리스트를 의미하고 연결리스트 맨 앞에 노드 주소를 head에 저장하고 연결리스트의 전체 사이즈를 count에 저장하였다.

해시 테이블 구조



- 따라서 해시 테이블 구조는 왼쪽 그림과 같다.

해시 함수

```
23
24 int getKey(char eng[ARR_SIZE])
25 {
26         int key = (strlen(eng) * (eng[0] - 'A')) % BUCKET_SIZE;
27         return key;
28 }
29
```

- 키를 구하는 방식은 왼쪽 함수로 만들었다.
- 영어 단어 길이와 영어단어 맨 첫글자 아스키 코드에서 ‘A’아스키 코드 뺀 값을 곱한 후 전체 버킷 사이즈로 나눈 값을 키값으로 사용하였다.

단어 추가하기

```
30
31 Node *createNode(char word[ARR_SIZE]){
32         Node* node;
33         node = (Node *)malloc(sizeof(Node));
34         if (node == NULL)
35                 return 0;
36         int i = 0;
37         int j = 0;
38         for (i = 0; i < ARR_SIZE; i++)
39         {
40                 node->eng[i] = '\0';
41                 node->kor[i] = '\0';
42         }
43         i = 0;
44         while (word[i] != ':')
45                 node->eng[j++] = word[i++];
46         node->eng[--j] = '\0';
47         j = 0;
48         i++;
49         while (word[i] != '\n' && word[i])
50                 node->kor[j++] = word[++i];
51         node->kor[--j] = '\0';
52         node->key = getKey(node->eng);
53         node->next = NULL;
54         return node;
55 }
```

- 단어를 추가할 때 우선 한개의 단어를 하나의 노드로 만들어야 하기 때문에 왼쪽 함수를 정의 하였다.
- 영어와 한글이 합쳐진 단어 전체를 받으면 그것을 분리하여 각각 node->eng, node->kor 에 저장하고 node->eng를 getKey에 보내 key값도 해당 노드에 저장하고 node->next를 NULL로 초기화 한뒤 node를 반환한다.

단어 추가하기

```
56
57 void add(char word[ARR_SIZE]){
58     Node* newNode = createNode(word);
59     int hashIndex = newNode->key;
60     if (hashTable[hashIndex].count == 0){
61         hashTable[hashIndex].count = 1;
62         hashTable[hashIndex].head = newNode;
63     }
64     else{
65         newNode->next = hashTable[hashIndex].head;
66         hashTable[hashIndex].head = newNode;
67         hashTable[hashIndex].count++;
68     }
69 }
```

- Main에서는 add함수만 호출하여 단어를 추가한다.
- 여기서도 한글과 영어가 합쳐진 한 단어 전체 문장을 매개변수로 받아와 바로 createNode로 보내 node 형태로 변환한 뒤 key값으로 해시테이블에서 자리를 찾아 해당 자리가 비어있으면 head자리에 넣고 비어있지 않으면 새 노드를 맨 앞에 넣고 그 뒤에 원래 있던 단어를 연결해준다.

단어 검색

```
70
71 void search(char eng[ARR_SIZE]){
72     int hashIndex = getKey(eng);
73     Node* node = hashTable[hashIndex].head;
74     int flag = 0;
75     int compare = 0;
76     while (node != NULL)
77     {
78         if (strcmp(node->eng,eng) == 0){
79             flag = 1;
80             break;
81         }
82         compare++;
83         node = node->next;
84     }
85     if (flag == 1){
86         printf("(%d회 검색)  %s\n", compare, node->kor);
87     }
88     else{
89         printf("\n 해당 단어가 존재 하지 않습니다. \n");
90     }
91 }
92
```

- 단어 검색은 영어 글자만 넘겨 받아 먼저 해시 키값을 계산한 후 해당 연결리스트를 찾아
- 그 연결리스트를 처음 부터 검색하면서 단어를 찾는다.

문제 2 – 실행 결과.

- 시작점과 도착점이 0, 3일때, 경로와 거리를 출력한다.

문제 2 코드 설명

```c
#include <stdio.h>
#include <limits.h>
#define TRUE 1
#define FALSE 0
#define MAX_VERTICES 7
#define INF 1000

int weight[MAX_VERTICES][MAX_VERTICES] = {
        { 0, 4, 8, INF, INF, INF, 10 },
        { 4, 0, 2, 5, 11, INF, INF},
        { 8, 2, 0, INF, 9, 4, 5},
        { INF, 5, INF, 0, 7, INF, INF},
        { INF, 11, 9, 7, 0, 2, 8},
        { INF, INF, 4, INF, 2, 0, INF},
        { 10, INF, 5, INF, 8, INF, 0}};
int distance[MAX_VERTICES];
int found[MAX_VERTICES];
int path[MAX_VERTICES][MAX_VERTICES];
int check[MAX_VERTICES];
```

- 코드는 수업시간에 배운 ppt에 나온 소스코드를 참고하였으며,
- 경로 출력을 위해 경로를 따로 저장하는 path배열과 한 정점으로 가는 정점을 저장하는 check 배열을 추가로 선언하였다.

Path 초기화

- Path 이중 배열을 INF로 초기화 해주는 함수이다.

```
21 void path_init(int path[][MAX_VERTICES]){
22         int i,j;
23         for(i=0;i<MAX_VERTICES;i++)
24                 for(j=0;j<MAX_VERTICES;j++)
25                         path[i][j] = INF;
26 }
```

최단경로 알고리즘

```
43 void shortest_path(int start, int n)
44 {
45         int i, j, u, w;
46         for (i = 0; i < n; i++)
47         {
48                 distance[i] = weight[start][i];
49                 found[i] = FALSE;
50                 check[i] = 1;
51                 path[i][0] = start;
52         }
53         found[start] = TRUE;
54         distance[start] = 0;
55         for (i = 0; i<n-2; i++)
56         {
57                 u = choose(distance, n, found);
58                 found[u] = TRUE;
59                 for (w = 0; w < n; w++)
60                 {
61                         if(!found[w])
62                         {
63                                 if (distance[u]+weight[u][w] < distance[w])
64                                 {
65                                         if (i==0)
66                                         {
67                                                 path[w][check[w]] = u;
68                                                 check[w]++;
69                                         }
70                                         else{
71                                                 for(j = 0; j<(check[u]+1); j++)
72                                                 {
73                                                         path[w][j] = path[u][j];
74                                                         path[w][j+1] = u;
75                                                         check[w]++;
76                                                 }
77                                         }
78                                         distance[w] = distance[u] + weight[u][w];
79                                 }
80                         }
81                 }
82         }
83 }
84
```

- 최단 경로 알고리즘은 수업 시간에 배운 내용을 그대로 가져왔으며
- 경로만 path함수에 따로 저장하였다.
- Path함수에는 시작점에서 모든 정점까지의 경로가 전부 들어가있다.

Main 함수

```
 85 int main()
 86 {
 87         int i, j, start, end;
 88
 89         path_init(path);
 90         printf("시작점?: ");
 91         scanf("%d", &start);
 92         printf("도착점?: ");
 93         scanf("%d", &end);
 94         shortest_path(start, MAX_VERTICES);
 95         printf("경로는 ");
 96         for (j = 0; j < MAX_VERTICES; j++){
 97                 if(path[end][j]!=INF){
 98                         printf("%d-",path[end][j]);
 99                 }
100         }
101         printf("%d\n",end);
102         printf("거리는 %d\n",distance[end]);
103 }
~
```

- Main함수에서 시작점과 도착점 을 입력 받아
- 시작점을 기준으로 다익스트라 알고리즘을 돌린 후
- 경로는 path 이중 배열에서 도착점에 해당하는 경로만 따로 출력해준다.