자료구조 프로그래밍 과제 4

제출일: 2022-12-12
학번/이름: 2016121150 / 윤준영

문제 1. Dijkstra 알고리즘을 사용하여, 0번 노드에서 다른 모든 노드들까지의 최단거리 거리를 계산하는 프로그램을 작성하시오.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define LINE_SIZE 16

int* loadfile(char* fn, int* n) {
    // input: file name, student struct array
    // output: number of student
    FILE* fp;
    char line[LINE_SIZE];
    char *lp;
    *n = 0;
    int a, b, c;
    if ((fopen_s(&fp, fn, "r")) != NULL) {
        printf("\nWrong File!");
        return NULL;
    } // return NULL, no file
    fgets(line, sizeof(line), fp);
    if (line[strlen(line)-1] == 10) line[strlen(line)-1] = 0;
    // remove \n if exist
    *n = atoi(line);
    int *cost = (int *)calloc((*n)*(*n), sizeof(int));
    // make cost matrix
    while(1) {
        fgets(line, sizeof(line), fp);
        if (line[strlen(line)-1] == 10) line[strlen(line)-1] = 0;
        // remove \n if exist
        //printf("%s\n", &line);
        lp = strtok(line, " ");     // get src
        a = atoi(lp);
        lp = strtok(NULL, " ");     // get dst
        b = atoi(lp);
        lp = strtok(NULL, " ");     // get cost
        c = atoi(lp);
        cost[a*(*n)+b] = c;
        if (feof(fp)) break;
    }
    fclose(fp);
    return cost;
}
```

1) loadfile: 입력받은 파일 이름을 통해 파일을 열고, 배열의 크기 n을 변경하고, cost 배열을 만들어 주는 함수이다.
fgets 함수를 통해 txt파일에 저장된 문자를 한 줄씩 읽어 strtok을 이용하여 공백으로 구분한 뒤 첫 줄은 정수로 변환하여 n에 저장하고, 다음은 a, b, c로 구분하여 cost[a][b]에 정수 c로 저장한다. 과제 3 문제 1의 loadfile 함수를 일부 수정하여 사용하였다.

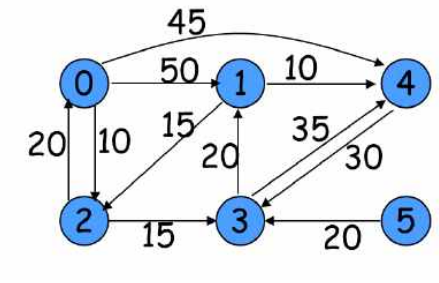
```
67  int* getcost(int* cost, int n, int s){
68      int* found = (int *)calloc(n, sizeof(int));
69      int* dist = (int *)calloc(n, sizeof(int));
70      found[s] = 1;
71      int j;
72      for (j = 0; j < n; j++) dist[j] = cost[j+s*n];
73      int sf = 0;
74      int osf = 0;
75      int min;
76      int i;
77      while (sf < n) {
78          // while sum of found < n
79          min = INT_MAX;
80          i = -1;
81          for (j = 0; j < n; j++)
82          // find minimum dist in which found is false
83              if (!found[j] && dist[j] && dist[j] <= min){
84              // if found is false and dist is not 0 and minimum, update min and index i
85                  min = dist[j];
86                  i = j;
87              }
88          if (i != -1){
89              found[i] = 1;
90              for (j = 0; j < n; j++)
91                  if (cost[i*n+j] && s!=j && (!dist[j] || dist[i]+cost[i*n+j] < dist[j]))
92                  // if cost[i][j] don't exists, skip
93                  // if s = j, skip
94                  // if dist is 0(MAX), update
95                  // if dist[i]+cost[i][j] < dist[j], update
96                      dist[j] = dist[i] + cost[i*n+j];
97          }
98          osf = sf;
99          sf = 0;
100         for (j = 0; j < n; j++) sf += found[j];
101         if (osf == sf) break;
102         // update sum of found
103         // if found array is not changed, break
104     }
105     for (j = 0; j < n; j++) {
106         cost[j+s*n] = dist[j];
107         if (s!=j) {
108             if (dist[j]) printf("%d --> %d : %d\n", s, j, dist[j]);
109             else printf("%d --> %d : No path\n", s, j);
110         }
111     }
112     return cost;
113 }
```

2) getcost: cost 배열, vertex의 개수 n, 계산할 출발 노드 s를 입력으로 받아 출발 노드 s에서 다른 노드로 가는 최소 거리를 계산하는 함수이다. 계산된 dist 배열을 cost에 적용하여 반환한다.
먼저 found와 dist 배열을 calloc 함수를 통해 동적 할당하여 초기화한다. 자기 자신으로 가는 found[s]를 1로 설정하고, dist 배열을 입력받은 cost 배열의 n번째 행으로 초기화한다.
while 문을 이용하여 found의 합을 계산하는 sf가 열의 크기 n보다 작을 때까지(found가 모두 1이 될 때까지) 반복한다. for과 if문을 이용하여 found[j]가 0인 j 중 dist[j]가 최소인 j를 찾아 i로 설정한다. 여기서 dist[j]가 존재하지 않았을 경우 0으로 설정하였기 때문에 dist[j]가 0이 아닌지의 여부도 조건에 포함했다. i가 존재할 경우(초기화한 값 -1이 아닐 경우), j=0~n-1에 대하여 다음을 반복한다. 만약 cost[i][j]가 존재하지 않거나, 출발 노드 s와 j가 같으면 넘어가고, dist[j]가 0(MAX)로 설정되어 있거나, dist[i]+cost[i][j]가 dist[j]보다 작을 경우, dist[j]를 dist[i]+cost[i][j]로 변경한다. 저장되어 있는 sf를 osf로 옮기고 새로운 sf를 계산한다. 만약 osf가 sf와 같다면 sf의 변화가 생기지 않은 것이므로 while 문을 빠져나간다.
마지막으로 계산된 dist 배열을 cost 배열의 n번째 행으로 복사하며 출력한 뒤 cost를 반환한다.

```
115 int main(int argc, char* argv[]) {
116     int n;
117     int* cost = loadfile(argv[1], &n);
118     cost = getcost(cost, n, 0);
119     return 0;
120 }
```

3) main: loadfile 함수를 이용하여 입력받은 파일명을 이용하여 cost 배열과 vertex의 개수 n를 받아온다. getcost함수를 이용하여 0 노드에서 출발하는 최소값을 계산하여 cost를 갱신한다.

4) 실행 결과 : 우측 graph.txt를 입력으로 실행한 결과는 아래와 같다.



graph.txt

```
6
0 1 50
0 2 10
0 4 45
1 2 15
1 4 10
2 0 20
2 3 15
3 1 20
3 4 35
4 3 30
5 3 20
```

```
C:\Users\Oscar\Desktop\GitHub\2202DS>HW\PHW4\hw4_2016121150.exe graph.txt
0 --> 1 : 45
0 --> 2 : 10
0 --> 3 : 25
0 --> 4 : 45
0 --> 5 : No path
```

main 함수의 getcost 함수를 다음과 같이 수정하면 txt 파일 뒤에 원하는 노드를 입력하여 cost를 계산할 수 있다.

```
119     cost = getcost(cost, n, atoi(argv[2]));
```

```
C:\Users\Oscar\Desktop\GitHub\2202DS>HW\PHW4\hw4_2016121150.exe graph.txt 1
1 --> 0 : 35
1 --> 2 : 15
1 --> 3 : 30
1 --> 4 : 10
1 --> 5 : No path

C:\Users\Oscar\Desktop\GitHub\2202DS>HW\PHW4\hw4_2016121150.exe graph.txt 2
2 --> 0 : 20
2 --> 1 : 35
2 --> 3 : 15
2 --> 4 : 45
2 --> 5 : No path
```

원본 그래프를 통해 살펴보면, 위 결과가 잘 계산된 것을 확인할 수 있다.
다음과 같이 그래프를 수정하여도 결과가 잘 나오는 것을 확인할 수 있다.

graph1.txt

```
5
0 1 20
0 2 10
0 4 35
1 2 15
1 4 10
2 0 20
2 3 15
3 1 20
3 4 25
4 3 30
```

```
C:\Users\Oscar\Desktop\GitHub\2202DS>HW\PHW4\hw4_2016121150.exe graph1.txt 0
0 --> 1 : 20
0 --> 2 : 10
0 --> 3 : 25
0 --> 4 : 30

C:\Users\Oscar\Desktop\GitHub\2202DS>HW\PHW4\hw4_2016121150.exe graph1.txt 1
1 --> 0 : 35
1 --> 2 : 15
1 --> 3 : 30
1 --> 4 : 10

C:\Users\Oscar\Desktop\GitHub\2202DS>HW\PHW4\hw4_2016121150.exe graph1.txt 2
2 --> 0 : 20
2 --> 1 : 35
2 --> 3 : 15
2 --> 4 : 40
```