1. 코드 분석 및 주석

#include <stdio.h>

#include <stdlib.h>

#define TRUE 1

#define FALSE 0

#define MAX\_VERTICES 8

#define INF 1000L

int weight[MAX\_VERTICES][MAX\_VERTICES] =

{ {0,3,INF,INF,INF,INF,INF,12},

{3,0,8,INF,INF,INF,INF,10},

{INF,8,0,15,2,INF,INF,INF},

{INF,INF,15,0,INF,INF,INF,INF},

{INF,INF,2,INF,0,9,4,5},

{INF,INF,INF,INF,9,0,INF,INF},

{INF,INF,INF,INF,4,INF,0,6},

{14,10,INF,INF,5,INF,6,0} };

int selected[MAX\_VERTICES];

int parent[MAX\_VERTICES];

struct heap\_node {//거리와 원래 키 값을 보존하기 위한 구조체 선언

int vertices;

int key;//거리

}dist[MAX\_VERTICES+1];

void build\_min\_heap(void) {

int i = 1;

int length = MAX\_VERTICES;

int child, parent;

int temp;

while (i != length / 2) {

child = i \* 2;

if (dist[child].key < dist[child - 1].key) {

child = child + 1;

}

if (dist[i - 1].key < dist[child - 1].key) {

dist[MAX\_VERTICES] = dist[i - 1];

dist[i - 1] = dist[child - 1];

dist[child - 1] = dist[MAX\_VERTICES];

}

i++;

}

}

int Extract\_Min() {//정렬값이 0인 걸 return 한다.

int min;

int i;

for (i = 0; i < MAX\_VERTICES; i++) {

if ((selected[i] == FALSE)) {//queue에서 제거되지 않고 가장 작은 최소의 수

min = dist[i].vertices;

break;

}

}

return min;

}

void Decrease\_key\_min\_heap(int i, int key) {//i에 있는 key값을 더 작게 만들도록

if (key >= dist[i].key)

printf("error new key is not smaller thatn current key\n");

dist[i].key = key;

int parent;

while (i != 1) {//i=1이면 가장 높은 노드이기 때문에

parent = i / 2;//parent노드를 찾고

if (dist[i].key < dist[parent].key) {//만약 parent보다 작은 경우에는 바꿔준다

dist[MAX\_VERTICES] = dist[i];

dist[i] = dist[parent];

dist[parent] = dist[MAX\_VERTICES];

}

i = parent;//그리고 비교대상을 위로 올려준다.

}

}

void prim(int s, int n)

{

int i, u, v,j;

for (u = 0; u < n; u++) {

dist[u].key = INF;

dist[u].vertices = u;

selected[u] = FALSE;

}

dist[s].key = 0;

for (i = 0; i < n; i++) {

u = Extract\_Min;//dist heap에서 가장 작은 것의 vertices 값을 추출

selected[u] = TRUE;//그 vertice에 true저장.

printf("%d", u);

for (v = 0; v < n; v++) {

if (weight[u][v] != INF)

if (!selected) {

for (j = 0; j < MAX\_VERTICES; j++) {//heap node안에 있는 vertices가 v인것을 찾아내야함.

if (dist[j].vertices == v)

break;

}

if (weight[u][v] < dist[j].key) {

Decrease\_key\_min\_heap(j, weight[u][v]);//현재 heap 에서 j번째로 정렬되어있는 것을 바꿔준다.

parent[v] = u;

}

}

}

}

}

void print\_prim() {

int i;

for (i = 0; i < MAX\_VERTICES; i++) {

printf("Vertices %d --> parent %d\n", dist[i].vertices, parent[dist[i].vertices]);

}

}

int main() {

prim(0,MAX\_VERTICES);

print\_prim();

}

1. 결과 및 콘솔 창

최선을 다하여 디버깅을 해보았지만 출력이 불가능했다.

1. 문제점

내 지식 수준에서는 문제점을 발견하지 못했다. Dist가 새로 업데이트 될 때 마다 정렬을 해주는 decrese min heap을 사용했고 또한 정렬된 순서와 vertices가 통합되지 않는 것을 염려하여 heap node를 선언하고 그 안에 vertice 값을 저장해서 코드를 짰지만 역시 출력이 되지 않았다. 어디서부터 출력이 막히는지 알아보기 위해서 일일히 printf를 해보았는데 Extract\_min 이 부분에서부터 출력이 막히는 것 같아. 하지만 그 이유는 잘 모르겠다.

내 추측으로 보건데 inf 부분이 비교가 안되는 것이 아닐까 생각해본다. 코드를 작성하기 위해서 몇 날 며칠을 매달렸지만 결국 풀지 못해 너무 아쉽다.