REPORT

(알고리즘 3차 과제)



|  |  |
| --- | --- |
| 제목 | 프림MST |
| 제출일자 | 20.10.23 |
| 소속학과 | 컴퓨터공학과 |
| 학번 | 2017305039 |
| 성명 | 신동민 |

문제1) 거스름돈에 대한 최소 동전수

#include <stdio.h>

void CoinchangeA(int change) {

int Coin[] = { 500,100,50,10,1 }; // 각 코인단위와 개수를 저장할 배열선언

int Count[] = { 0,0,0,0,0,0,0 };

while (change >= 500) { //거스름돈이 코인단위 값보다 높으면 단위만큼 빼고 개수를 증가

change = change - 500;

Count[0]++;

}

while (change >= 100) {

change = change - 100;

Count[1]++;

}

while (change >= 50) {

change = change - 50;

Count[2]++;

}

while (change >= 10) {

change = change - 10;

Count[3]++;

}

while (change >= 1) {

change = change - 1;

Count[4]++;

}

printf("규정 A: ");

for (int i = 0; i < 5; i++) { //코인단위에 따른 거스름돈개수 출력

printf("%d원:%d개 ", Coin[i], Count[i]);

}

printf("\n");

}

void CoinchangeB(int change) {

int Coin[] = { 500,130,100,50,10,5,1 }; // 각 코인단위와 개수를 저장할 배열선언

int Count[] = { 0,0,0,0,0,0,0 };

while (change >= 500) { //거스름돈이 코인단위 값보다 높으면 단위만큼 빼고 개수를 증가

change = change - 500;

Count[0]++;

}

while (change >= 130) {

change = change - 130;

Count[1]++;

}

while (change >= 100) {

change = change - 100;

Count[2]++;

}

while (change >= 50) {

change = change - 50;

Count[3]++;

}

while (change >= 10) {

change = change - 10;

Count[4]++;

}

while (change >= 5) {

change = change - 5;

Count[5]++;

}

while (change >= 1) {

change = change - 1;

Count[6]++;

}

printf("규정 B: ");

for (int i = 0; i < 7; i++) { //코인단위에 따른 거스름돈개수 출력

printf("%d원:%d개 ", Coin[i], Count[i]);

}

printf("\n");

}

void main() {

int change;

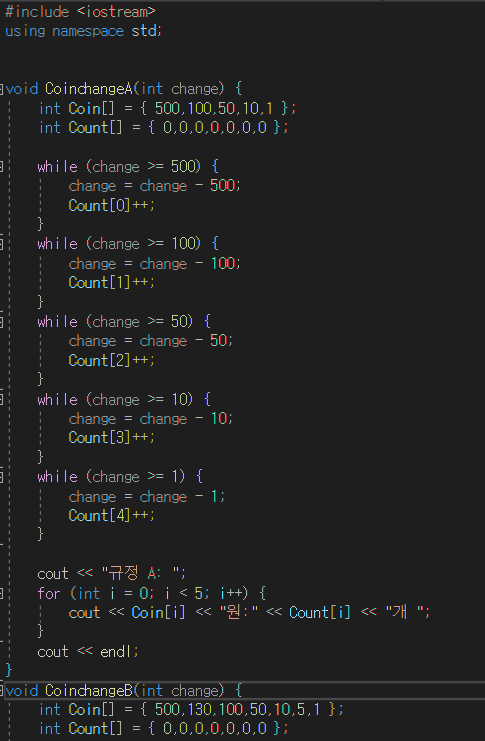
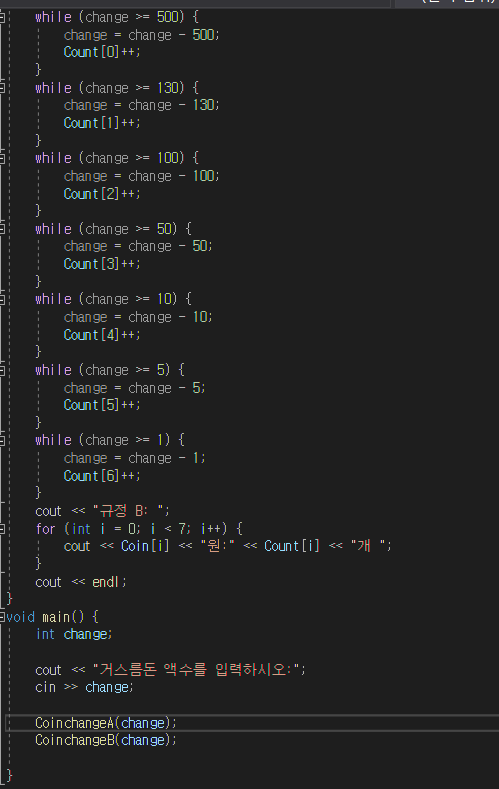
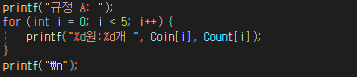
printf("거스름돈 액수를 입력하시오:"); //받아야하는 거스름돈을 입력

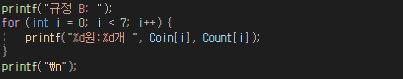
scanf\_s("%d", &change);

CoinchangeA(change); //규정A에 맞는 거스름돈함수

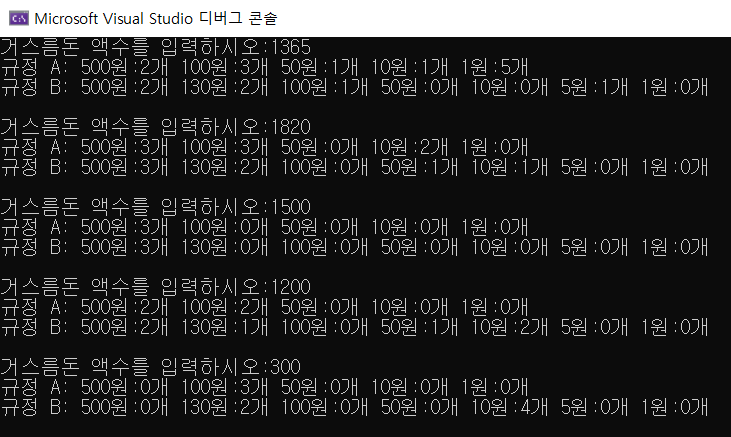
CoinchangeB(change); //규정b에 맞는 거스름돈함수

}









문제2) Prim적용 최소신장트리

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define MAX 26 //무한대를 정의

#define INF 1000L

int Matrix[MAX][MAX] = { 0 }; ` //인접행렬의 배열

int dist[MAX]; //시작점과 다른 점들의 가중치저장

int from[MAX];

void GraphMaker() { //그래프를 인접행렬방식으로 생성

srand((unsigned)time(NULL));

int Tindex = 0; //4개의 간선을 랜덤하게 받은 열의 인덱스

int Weight = 0; //가중치변수

int count1 = 0; //가로줄 가중치 개수 저장

int count2 = 0; //세로줄 가중치 개수 저장

for (int i = 0; i < MAX; i++) { //각 점을 제외하고 무한대로 입력

for (int j = 0; j < MAX; j++)

if (i != j)

Matrix[j][i] = INF;

}

for (int i = 0; i < MAX; i++) {

while (true) {

if ((MAX - i - 1) > 4) {

count1 = 0;

count2 = 0;

for (int j = 0; j < MAX; j++) { //가로열에 대한 가중치 개수 검사

if (Matrix[i][j] != INF && Matrix[i][j] != 0)

count1++;

}

if (count1 > 3) //간선이 4개면 간선을 그만 늘린다.

break;

else { //간선이 부족하다면 채운다.

for (int j = 0; j < 4 - count1; j++) {

Tindex = (rand() % (MAX - i - 1)) + i + 1;

for (int k = 0; k < MAX; k++) {

if (Matrix[k][Tindex] != INF && Matrix[k][Tindex] != 0)

count2++;

}

if (count2 > 3)

break;

Weight = (rand() % 99) + 1 //가중치 값을 랜덤으로 받고

Matrix[i][Tindex] = Weight //무방향그래프로 만든다.

Matrix[Tindex][i] = Weight;

}

}

}

else { //범위가 다를 뿐 위에 작업 똑같이 한다.

count1 = 0;

count2 = 0;

for (int j = 0; j < MAX; j++) {

if (Matrix[i][j] != INF && Matrix[i][j] != 0)

count1++;

}

if (count1 > 3)

break;

else {

for (int j = 0; j < MAX - i - 1 - count1; j++) {

Tindex = (rand() % (MAX - i - 1)) + i + 1;

for (int k = 0; k < MAX; k++) {

if (Matrix[Tindex][k] != INF && Matrix[Tindex][k] != 0)

count2++;

}

if (count2 > 3)

break;

Weight = (rand() % 99) + 1; //가중치 값을 랜덤으로 받고

Matrix[i][Tindex] = Weight; //무방향그래프로 만든다.

Matrix[Tindex][i] = Weight;

}

}

}

break;

}

}

for (int i = 0; i < MAX; i++) { //입력 받은 인접행렬을 출력

for (int j = 0; j < MAX; j++) {

if (Matrix[i][j] == INF)

printf("INF ");

else

printf("%3d ", Matrix[i][j]);

}

printf("\n");

}

printf("\n");

}

int OutMin() { //시작점,간선으로 이어진 점들과 이어지지 않은 점들의 가중치가 제일 작은 점을 반환함수

int min = INF;

int best = INF;

for (int i = 0; i < MAX; i++) { //가중치가 가장 작은 인덱스를 구하는 함수

if (dist[i] != 0 && dist[i] < min) {

min = dist[i];

best = i;

}

}

return best;

}

void PrimMST(int start){ //프림 방식으로 최소 신장 트리를 구현한 함수

int T[2][MAX] = { 0 }; //이어진 점들을 차례대로 저장하는 배열

T[0][0] = start;

T[1][0] = start;

for (int i = 0; i < MAX; i++) { //시작점의 가중치들을 설정

dist[i] = Matrix[start - 1][i];

from[i] = start;

}

for (int i = 0; i < MAX - 1; i++) { //간선을 만드는 작업

int best = OutMin(); //가중치가 최소인 인덱스를 구하는 함수

if (best == INF)return;

T[0][i + 1] = from[best];

T[1][i + 1] = best + 1;

dist[best] = 0;

for (int j = 0; j < MAX; j++) { //시작점과 이어진 점들의 가중치들을 비교하여 갱신

if (Matrix[best][j] != 0 && Matrix[best][j] < dist[j]) {

dist[j] = Matrix[best][j];

from[j] = best + 1;

}

}

}

for (int i = 0; i < MAX; i++) { //시작점과 간선을 출력

if (i == 0)

printf("시작점: %d-%d\n", T[0][i], T[1][i]);

else

printf("간선%d: %d-%d\n", i, T[0][i], T[1][i]);

}

}

int main()

{

int start; //시작점을 입력 받는다.

printf("시작점을 입력하세요: ");

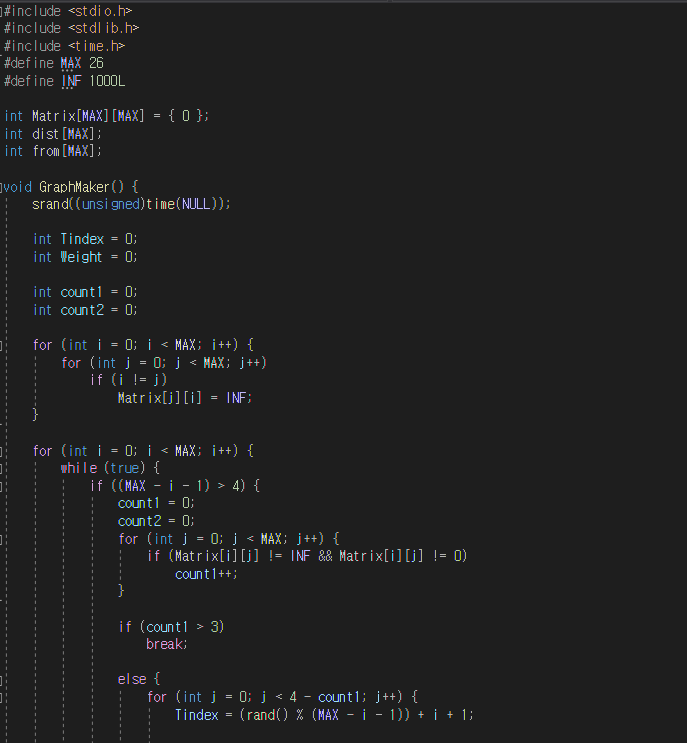
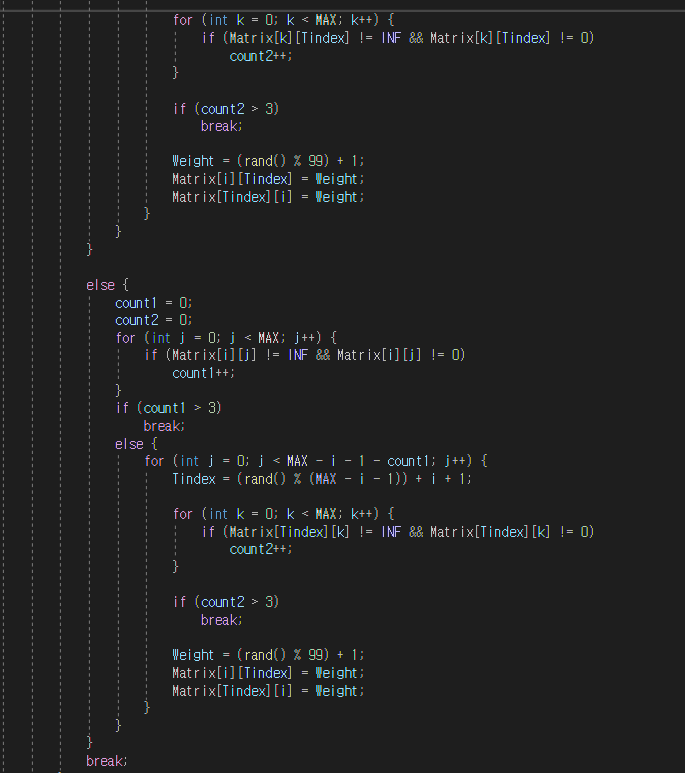
scanf\_s("%d", &start);

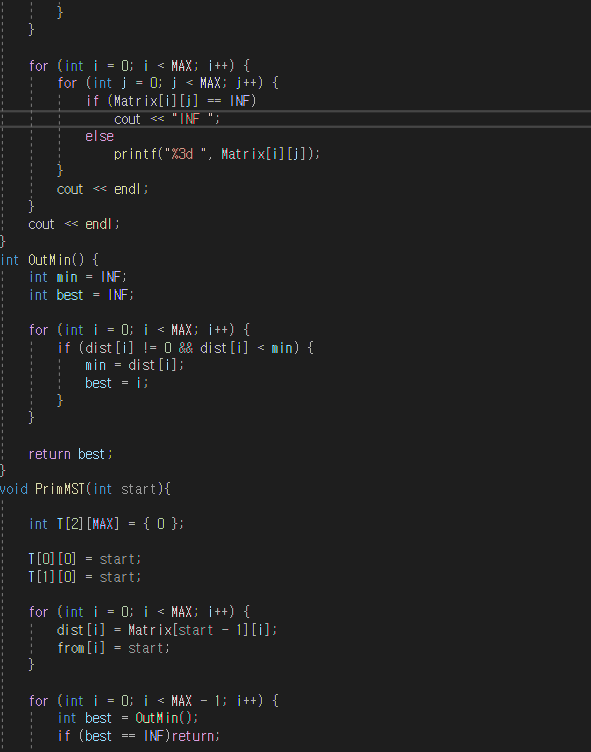
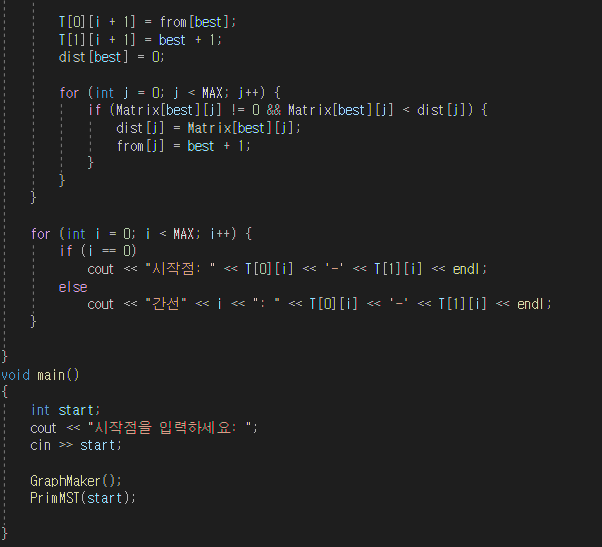
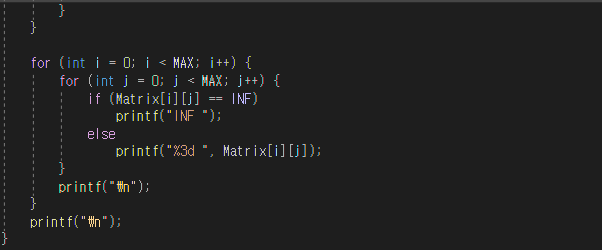
GraphMaker(); //무방향그래프 생성함수

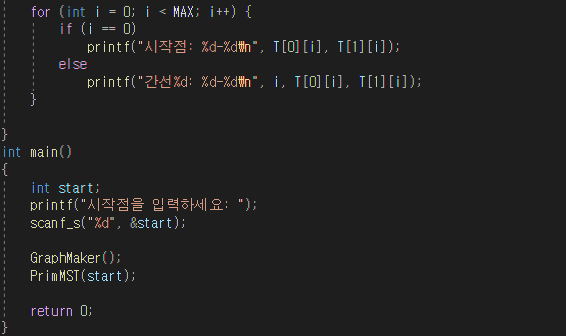
PrimMST(start); //최소신장트리 함수

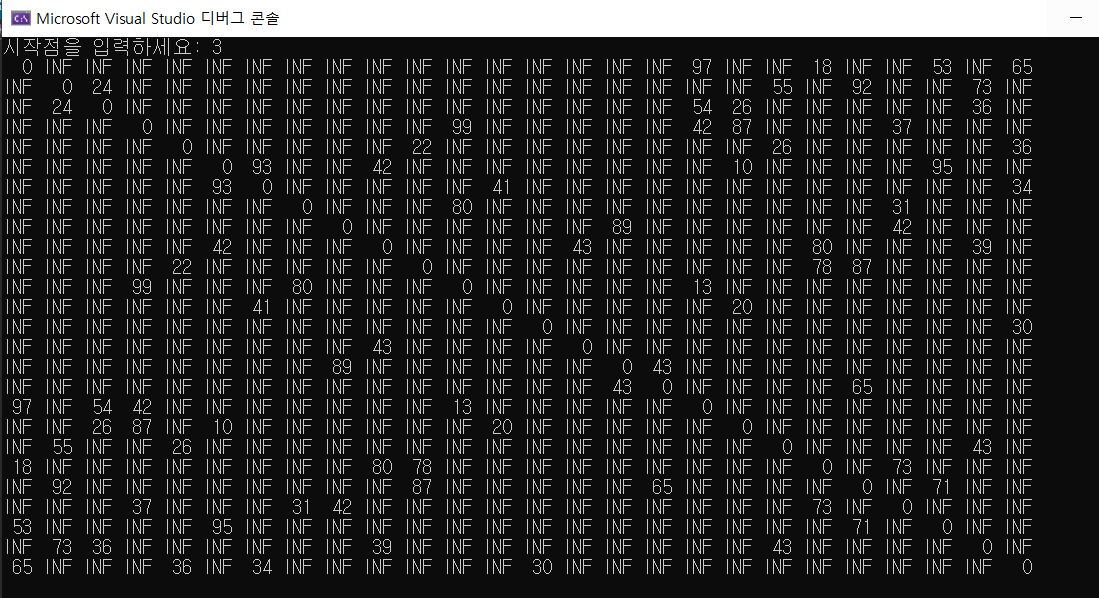
return 0;

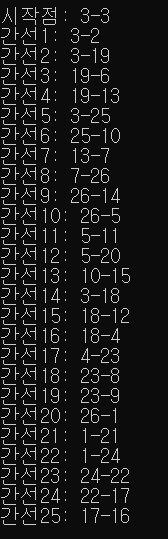
}











소감문

-문제 1번은 알고리즘 자체가 명료해서 프로그래밍하는 것이 쉬웠습니다. 먼저 받을 거스름돈을 입력 받아 규정A와 규정B에 맞게 거스름돈 개수를 출력하는데 규정B의 코인단위가 다양하게 되어 있어서 최적해를 더 잘 찾을 것 같지만 그렇지 않았습니다. 실제로 출력한 결과를 보면 1365,1820원은 규정A:12, 8 규정B:6, 7개로 규정B가 효율적이지만 1200,300원에서는 규정A가 좋습니다. 이 차이는 액명가별 기준과 거스름돈에 따라 최적해를 찾는데 영향을 준다는 것을 알았으며 최적해를 못 찾았을 수도 있다고 생각했습니다.

문제 2번은 프림알고리즘으로 MST을 표현하는 프로그램인데 트리를 만들기전에 필요한 무방향그래프를 먼저 설계해야 합니다. 그래프를 설계방식으로 인접행렬을 사용했는데 자료구조를 배우지 않아서 정보를 찾아 개념공부를 하고 만드는데 시간이 많이 걸렸습니다. 랜덤으로 그래프의 점당 최대 4개의 간선을 가지게 했으며 중복으로 간선이 생기면 3개이하가 되고 무조건 1개이상의 간선이 이어지게 설계했으며 가중치는 100미만의 값을 랜덤으로 넣었습니다. 그리고 각 점의 행렬에는 0값을 점과 간선을 뺀 나머지에는 무한대인 INF를 정의해서 그래프를 완성시켰습니다. 프림알고리즘은 시작점인 start를 받아 dist배열에 다른 점들의 가중치를 입력 받고 최단거리 점의 인덱스를 받아 그 점을 T배열에 넣어 간선으로 이어준 것처럼 표현했으며 dist의 가중치와 이어진 점과의 가중치를 갱신하여 프림알고리즘을 완성시켰습니다. 강의로 알고리즘만 들었을 때에는 어떻게 C언어로 표현하지 생각으로 걱정했는데 막상 들었던 알고리즘을 그대로 작성하니 실행되는 것을 보고 어려운 문제라도 알고리즘만 짤 수 있으면 쉽게 풀 수 있다고 생각했습니다.