

알고리즘 2021 보고서

보고서 제출서약서

나는 송실대학교 컴퓨터학부의 일원으로 명예를 지키면서 생활하고 있습니다.

나는 보고서를 작성하면서 다음과 같은 사항을 준수하였음을 엄숙히 서약합니다.

1. 나는 자력으로 보고서를 작성하였습니다.
 - 1.1. 나는 동료의 보고서를 베끼지 않았습니다.
 - 1.2. 나는 비공식적으로 얻은 해답/해설을 기초로 보고서를 작성하지 않았습니다.
2. 나는 보고서에서 참조한 문헌의 출처를 밝혔으며 표절하지 않았습니다. (나는 특히 인터넷에서 다운로드한 내용을 보고서에 거의 그대로 복사하여 사용하지 않았습니다.)
3. 나는 보고서를 제출하기 전에 동료에게 보여주지 않았습니다.
4. 나는 보고서의 내용을 조작하거나 날조하지 않았습니다.

과목	알고리즘 2021
과제명	Traveling SalesPerson Program (가중치를 포함한 그래프)
담당교수	최 재 영 교 수
제출인	컴퓨터학부 20162449 김상현 [307]
제출일	2021년 10월 22일

차 례

1장 프로젝트 동기/목적

2장 설계/구현 아이디어

3장 수행결과(구현 화면 포함)

4장 결론 및 보충 할 점

5장 디버깅

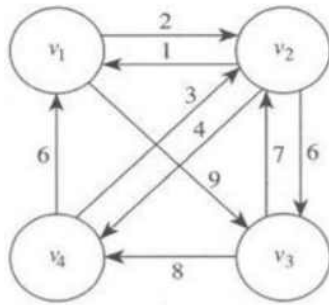
6장 소스코드(+주석)

1. 프로젝트 동기 및 목적

Traveling Salesperson Problem을 주어진 수도 코드와 가중치 포함 그래프를 이용해서 구현하고 풀어본다.

2. 설계 구현 아이디어 (2번 문제)

주어진 가중치 포함 그래프를 프로그래밍 상에서 구현하기 위해서 행렬로 표현한다.



변환 후 인접행렬로 나타냄 ->

```
int W[N][N] = {
    {0, 2, 9, INF},
    {1, 0, 6, 4},
    {INF, 7, 0, 8},
    {6, 3, INF, 0}
}; //인접행렬 입력값
```

수도 코드 상에서 인자로 주어지는 값들을 전역변수와 사용자 정의값으로 정리한다.

```
#define N 4 //정점의 수
#define M 16 //부분집합의 수 '2의 N승'
#define VS 1 //시작경로
#define VSP (int)pow(2, VS-1)
#define INF 10000
```

```
int D[N][M]; //최단경로 저장배열
int P[N][M]; //P[vi][A] = D[vi][A]의 최단경로로 갈때 처음가는 원소
int set[M]; //각 부분집합이며, 원소갯수를 저장한다.
```

travel() 함수를 dynamic 프로그래밍을 이용하여 구현한다고 생각했을 때, 기본 적인 알고리즘은 아래와 같다.

$D[1][\{2,3,4\}]$ = (v1에서 출발하여 모든 노드를 지나고 최소 비용으로 v1으로 돌아오는 수)
= $\min(W[1,2] + D[2][\{3,4\}], W[1,3]+D[\{2,4\}], W[1,4]+D[\{2,3\}])$

Tree 로 나타낸다면 굉장히 복잡하기 때문에 한가지 경우만 예를 들어 설명해 보려고 한다.

$W[1,2] + D[2][\{3,4\}]$ = $2 + D[2][\{3,4\}]$
= $2 + \{ \min(W[2,3] + D[3][\{4\}], W[2,4]+D[4][\{3\}]) \}$

$W[2,3] + D[3][\{4\}]$ = $6 + D[3][\{4\}]$
= $6 + W[3,4] + D[4][x(\text{공집합})]$

이런 식으로 dynamic programming을 설계 할 수 있다.

교재에 주어진 수도코드를 참고하여 구현한 travel() 함수는 아래와 같다.

```
int travel() {
    //외판원 경로구하기
    int i, k, A;
    setCount();

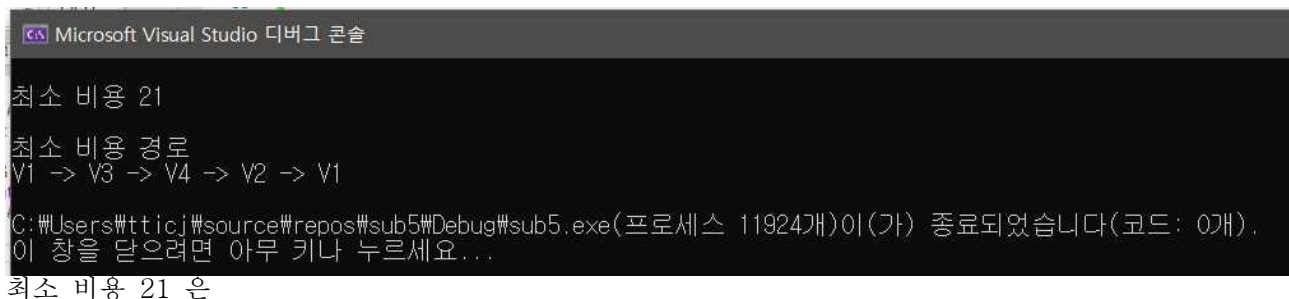
    printf("\n");
    int minlength;

    for (i = 0; i < N; i++)
        D[i][0] = W[i][VS - 1];

    for (k = 1; k <= N - 2; k++) {
        for (A = 0; A < M; A++) {
            if (set[A] == k && (A & VSP) == 0) {
                for (i = 0; i < N; i++) {
                    if (A & (int)pow(2, i))
                        continue;
                    min(i, A); //min(W[i][j]+D[j][A-j])
                }
            }
        }
    }

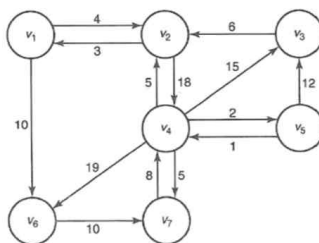
    min(VS - 1, M - 1 - VSP);
    // 노드의 index 는 0 부터 시작이지만 노드의 번호는 1부터 시작이기에 - 1을 해주는 것을 잊지말자
    minlength = D[VS - 1][M - 1 - VSP];
    return minlength; //최소비용 리턴
}
```

3. 수행결과 (문제 2번 / 문제 3번)



PDF 로 첨부된 1번 문제에서 해결한 최소 비용 21과 같다.

문제 3번에서 주어진 가중치 포함 그래프는 아래와 같다.



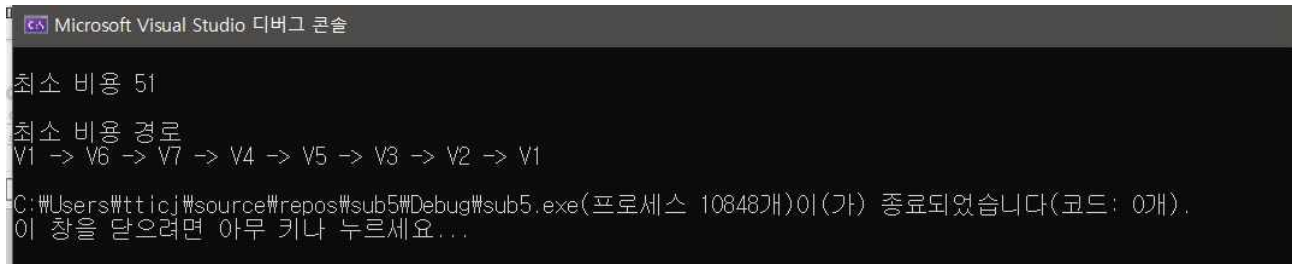
-> 행렬로 표현하면 ->

```
int W[N][N] = {
    {0, 4, INF, INF, INF, 10, INF},
    {3, 0, INF, 18, INF, INF, INF},
    {INF, 6, 0, INF, INF, INF, INF},
    {INF, 5, 15, 0, 2, 19, 5},
    {INF, INF, 12, 1, 0, INF, INF},
    {INF, INF, INF, INF, INF, 0, 10},
    {INF, INF, INF, 8, INF, INF, 0}
};
```

물론 기존의 사용자 정의형으로 정의한 N과 M 값 또한 변경해준다.

```
#define N 7  
#define M 128  
.....
```

수행 결과는



```
Microsoft Visual Studio 디버그 콘솔  
최소 비용 51  
최소 비용 경로  
V1 -> V6 -> V7 -> V4 -> V5 -> V3 -> V2 -> V1  
C:\Users\wticj\source\repos\sub5\Debug\sub5.exe(프로세스 10848개)이(가) 종료되었습니다(코드: 0개).  
이 창을 닫으려면 아무 키나 누르세요...
```

최소 비용 51 ,

V1 ->V6 -> V7 -> V4 -> V5 -> V3 -> V2 -> V1 의 최소 비용 경로를 가진다.

4. 결론 및 보충할 점

추가 하고 싶은 내용이 있어서 결론 부분에 보충해 보려고한다. 처음 구현할 때 생각했던 방법은 $D[][\{\}]$ 에 전달 되는 부분집합 부분을 비트 표현으로 해보려고 하였다.

즉 $D[v1][\{v2,v3,v4\}]$ 의 경우,

$D[v1][1 1 1 0 (2)]$ 으로 표현 하여 비트 마스킹을 통해 함수에 진입 하였을 때 부분집합을 구분 할 수 있게 하려고 하였다. 하지만 실력 부족으로 완벽하게 구현하지 못했고 결국 인터넷 서핑 등을 통해서 최대한 수도코드와 비슷한 방향성을 가지고 코딩하게 되었다. 다음번에 구현할 시간이 충분하다면 기존에 생각했던 방식으로도 도전해 보고싶다.

5. 디버깅

Debugger window showing the execution of a program. The main window displays the source code with a breakpoint at line 89. The right sidebar shows performance metrics: Events (0), Process Memory (100), and CPU (100). The bottom pane shows a variable watch window with variables i, j, and k.

이름	값	형식
i	1	int
j	1	int
k	0	int

Debugger window showing the execution of a program. The main window displays the source code with a breakpoint at line 67. The right sidebar shows performance metrics: Events (0), Process Memory (100), and CPU (100). The bottom pane shows a variable watch window with variables A, k, and set[A].

이름	값
A	1
k	1
set	0x0019ce30 {0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, ...}
set[A]	1

6. 소스 코드

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#define N 7
#define M 128
// #define N 4
////1번,2번 문제 정점의 수
// #define M 16
//부분집합의 수 '2의 N승'
#define VS 1 //시작경로
#define VSP (int)pow(2,VS -1)
#define INF 1000
int D[N][M]; //최단경로 저장배열
int P[N][M]; //P[Vi][A] = D[Vi][A]의 최단경로
로 갈때 처음가는 원소
int set[M]; //각 부분집합이며, 원소갯수를 저장
한다.
int W[N][N] = {
    {0,4,INF,INF,INF,10,INF},
    {3,0,INF,18,INF,INF,INF},
    {INF,6,0,INF,INF,INF,INF},
    {INF,5,15,0,2,19,5},
    {INF,INF,12,1,0,INF,INF},
    {INF,INF,INF,INF,INF,0,10},
    {INF,INF,INF,8,INF,INF,0}
};
//int W[N][N] = { // 1,2번 문제 인접행렬
//    {0,2,9,INF},
//    {1,0,6,4},
//    {INF,7,0,8},
//    {6,3,INF,0}
//};
//인접행렬 입력값
void setCount();
int travel();
void path(int, int);
void min(int, int);
int main(int argc, char * argv[])
{
    printf("최소 비용 %d WnWn", travel());
    //최소비용 출력
    printf("최소      비용      경로Wn");
    //경로출력
    path(VS -1, M -1 - VSP);
    return 0;
}
int travel() {
    //외판원 경로구하기
    int i, k, A;
    setCount();
    printf("Wn");
    int minlength;
    for (i =0; i < N; i++)
        D[i][0] = W[i][VS -1];
    for (k =1; k <= N -2; k++) {
        for (A =0; A < M; A++) {
```

```
if (set[A] == k && (A
& VSP) ==0) {
    for (i =0; i <
N; i++) {
        if (A
& (int)pow(2, i))
            continue;
        min(i,
A); //min(W[i][j]+D[j][A-j]
    }
    }
    }
    min(VS -1, M -1 - VSP);
    // 노드의 index 는 0 부터 시작이지만
    노드의 번호는 1부터 시작이기에 - 1을 해주는
    것을 잊지말자
    minlength = D[VS -1][M -1 - VSP];
    return minlength; //최소비용 리턴
}
void setCount() {
    //각 부분집합들의 원소 갯수를 세는 함
    수
    int A, I, J, K;
    for (A =0; A < M; A++) {
        J = A;
        I = M /2;
        K =0;
        while (J >0) {
            if (J - I >=0) {
                J = J - I;
                K++;
            }
            I /=2;
        }
        set[A] = K; //set[A]에 원
소의 갯수를 넣어줌
    }
}
void min(int i, int A) {
    int j, s;
    D[i][A] = INF; // ∞(INF)로 초기화
    for (j =0; j < N; j++) {
        s = (int)pow(2, j);
        if ((A & s) !=0) {
            if (D[i][A] > W[i][j]
+ D[j][A - s]) {
                D[i][A] =
W[i][j] + D[j][A - s];
                P[i][A] = j;
            }
        }
    }
}
void path(int i, int A) {
    int j = P[i][A];
```

```
printf("V%d -> ", i + 1);
if (A - pow(2, j) != 0)
    path(j, (int)A - pow(2, j));
    //다음 가야할 경로 호출
else
    printf("V%d -> V%dWn", j + 1,
VS); //도착
}
```