알고리즘 7차 과제

-TSP Algorithm-

컴퓨터학부 (나)반

20162527 (출석번호 : 251)

함인규

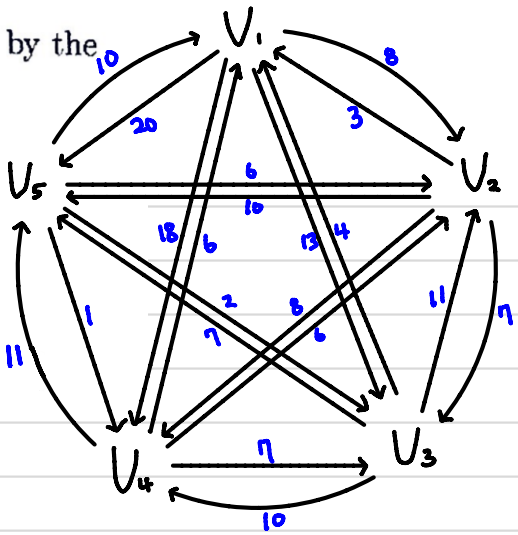
목차

28-29번 문제

28. Find an optimal circuit for the weighted, direct graph represented by the following matrix W. Show the actions step by step.

$$W = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 8 & 13 & 18 & 20 \\ 3 & 0 & 7 & 8 & 10 \\ 4 & 11 & 0 & 10 & 7 \\ 6 & 6 & 7 & 0 & 11 \\ 10 & 6 & 2 & 1 & 0 \end{bmatrix} \end{array}$$

그래프



①

$D[v_2][\emptyset] = W[2][1] = 3$

$D[v_3][\emptyset] = W[3][1] = 4$

$D[v_4][\emptyset] = 6$

$D[v_5][\emptyset] = 10$

② $k=1$

$D[v_2][\{v_3\}] = \min(W[3][2] + D[v_2][\emptyset]) = 11+3 = 14$ $j=2$

$D[v_4][\{v_2\}] = \min(W[4][2] + D[v_2][\emptyset]) = 6+3 = 9$ $j=2$

$D[v_5][\{v_2\}] = \min(W[5][2] + D[v_2][\emptyset]) = 6+3 = 9$ $j=2$

$D[v_2][\{v_3\}] = \min(W[2][3] + D[v_3][\emptyset]) = 7+4 = 11$ $j=3$

$D[v_4][\{v_3\}] = \min(W[4][3] + D[v_3][\emptyset]) = 7+4 = 11$ $j=3$

$D[v_5][\{v_3\}] = \min(W[5][3] + D[v_3][\emptyset]) = 2+4 = 6$ $j=3$

$D[v_2][\{v_4\}] = \min(W[2][4] + D[v_4][\emptyset]) = 8+6 = 14$ $j=4$

$D[v_3][\{v_4\}] = \min(W[3][4] + D[v_4][\emptyset]) = 10+6 = 16$ $j=4$

$D[v_5][\{v_4\}] = \min(W[5][4] + D[v_4][\emptyset]) = 1+6 = 7$ $j=4$

$D[v_2][\{v_5\}] = \min(W[2][5] + D[v_5][\emptyset]) = 10+10 = 20$ $j=5$

$D[v_3][\{v_5\}] = \min(W[3][5] + D[v_5][\emptyset]) = 7+10 = 17$ $j=5$

$D[v_4][\{v_5\}] = \min(W[4][5] + D[v_5][\emptyset]) = 11+10 = 21$ $j=5$

③ $k=2$

$D[V_4][\{V_2,V_3\}] = \min(W[4][2]+D[V_2][\{V_3\}],\ W[4][3]+D[V_3][\{V_2\}]) = \min(6+11,\ 7+14) = 17$ $j=2$

$D[V_5][\{V_2,V_3\}] = \min(W[5][2]+D[V_2][\{V_3\}],\ W[5][3]+D[V_3][\{V_2\}]) = \min(6+11,\ 2+14) = 16$ $j=3$

$D[V_3][\{V_2,V_4\}] = \min(W[3][2]+D[V_2][\{V_4\}],\ W[3][4]+D[V_4][\{V_2\}]) = \min(11+14,\ 10+9) = 19$ $j=4$

$D[V_5][\{V_2,V_4\}] = \min(W[5][2]+D[V_2][\{V_4\}],\ W[5][4]+D[V_4][\{V_2\}]) = \min(6+14,\ 1+9) = 10$ $j=4$

$D[V_3][\{V_2,V_5\}] = \min(W[3][2]+D[V_2][\{V_5\}],\ W[3][5]+D[V_5][\{V_2\}]) = \min(11+20,\ 7+9) = 16$ $j=5$

$D[V_4][\{V_2,V_5\}] = \min(W[4][2]+D[V_2][\{V_5\}],\ W[4][5]+D[V_5][\{V_2\}]) = \min(6+20,\ 11+9) = 20$ $j=5$

$D[V_2][\{V_3,V_4\}] = \min(W[2][3]+D[V_3][\{V_4\}],\ W[2][4]+D[V_4][\{V_3\}]) = \min(7+16,\ 8+11) = 19$ $j=4$

$D[V_5][\{V_3,V_4\}] = \min(W[5][3]+D[V_3][\{V_4\}],\ W[5][4]+D[V_4][\{V_3\}]) = \min(2+16,\ 1+11) = 12$ $j=4$

$D[V_2][\{V_3,V_5\}] = \min(W[2][3]+D[V_3][\{V_5\}],\ W[2][5]+D[V_5][\{V_3\}]) = \min(7+17,\ 10+6) = 16$ $j=5$

$D[V_4][\{V_3,V_5\}] = \min(W[4][3]+D[V_3][\{V_5\}],\ W[4][5]+D[V_5][\{V_3\}]) = \min(7+17,\ 11+6) = 17$ $j=5$

$D[V_2][\{V_4,V_5\}] = \min(W[2][4]+D[V_4][\{V_5\}],\ W[2][5]+D[V_5][\{V_4\}]) = \min(8+21,\ 10+7) = 17$ $j=5$

$D[V_3][\{V_4,V_5\}] = \min(W[3][4]+D[V_4][\{V_5\}],\ W[3][5]+D[V_5][\{V_4\}]) = \min(10+21,\ 7+7) = 14$ $j=5$

④ $k=3$

$D[V_5][\{V_2,V_3,V_4\}] = \min(W[5][2]+D[V_2][\{V_3,V_4\}],\ W[5][3]+D[V_3][\{V_2,V_4\}],\ W[5][4]+D[V_4][\{V_2,V_3\}])$
$= \min(6+19,\ 2+16,\ 1+17) = 18$ $j=3$

$D[V_4][\{V_2,V_3,V_5\}] = \min(W[4][2]+D[V_2][\{V_3,V_5\}],\ W[4][3]+D[V_3][\{V_2,V_5\}],\ W[4][5]+D[V_5][\{V_2,V_3\}])$
$= \min(6+16,\ 7+16,\ 11+16) = 22$ $j=3$

$D[V_3][\{V_2,V_4,V_5\}] = \min(W[3][2]+D[V_2][\{V_4,V_5\}],\ W[3][4]+D[V_4][\{V_2,V_5\}],\ W[3][5]+D[V_5][\{V_2,V_4\}])$
$= \min(11+17,\ 10+20,\ 7+10) = 17$ $j=5$

$D[V_2][\{V_3,V_4,V_5\}] = \min(W[2][3]+D[V_3][\{V_4,V_5\}],\ W[2][4]+D[V_4][\{V_3,V_5\}],\ W[2][5]+D[V_5][\{V_3,V_4\}])$
$= \min(7+14,\ 8+17,\ 10+12) = 21$ $j=3$

⑤ $k=4$

$D[V_1][\{V_2,V_3,V_4,V_5\}] = \min(W[1][2]+D[V_2][\{V_3,V_4,V_5\}],\ W[1][3]+D[V_3][\{V_2,V_4,V_5\}],$
$W[1][4]+D[V_4][\{V_2,V_3,V_5\}],\ W[1][5]+D[V_5][\{V_2,V_3,V_4\}])$
$= \min(8+21,\ 13+17,\ 18+22,\ 20+18) = 29$ $j=2$

Path

$P[V_1][\{V_2, V_3, V_4, V_5\}] = 2$

$P[V_2][\{V_3, V_4, V_5\}] = 3$

$P[V_3][\{V_4, V_5\}] = 5$

$P[V_5][\{V_4\}] = 4$

위의 결과에 따라
$V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_5 \rightarrow V_4 \rightarrow V_1$ 가 길이 29로 최적의 순환경로이다.

● 소스코드

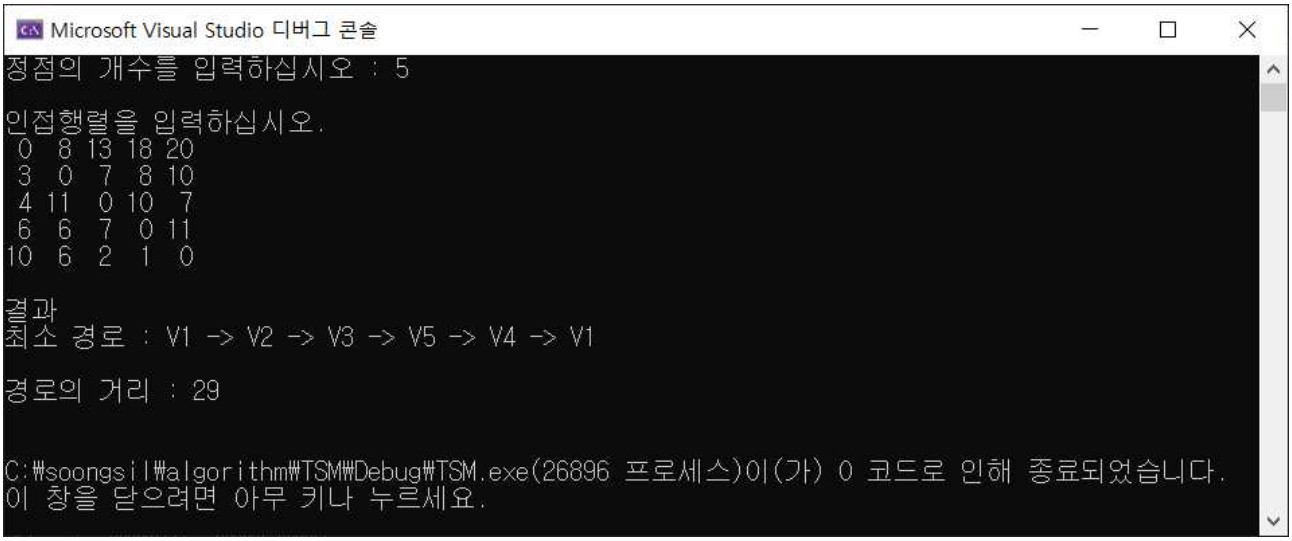
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


int** W;  //인접행렬 입력값
int** D;  //최단경로 저장배열
int** P;   //P[Vi][A] = D[Vi][A]의 최단경로로 갈때 처음가는 원소
int* subset;   //각 부분집합이며, 원소갯수를 저장한다.

int max;      //부분 집합 최대 원소 갯수
int node_EA;   //정점의 갯수


void mininum(int i, int A) {
	int j, s;
	//A는 부분 집합
	D[i][A] = 1000; // ∞(1000)로 초기화
	for (j = 0; j < node_EA; j++) {
		s = (int)pow(2, j);  //pow함수로 지수승 구하기 2^j
		if ((A & s) != 0) {  //Vj가 A에 있을 경우
			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;    //j의 최소값 저장
			}
		}
	}
}

int travel() {      //경로 구하기
	int i, k, A;
	int minlength; //최적 경로 길이

	for (i = 0; i < node_EA; i++)
		D[i][0] = W[i][0];

	for (k = 1; k <= node_EA - 2; k++)
	{
		for (A = 0; A < max; A++)
		{      //subset A는 자신을 뺀 나머지 k개의 정점을 포함한 부분집합
			if (subset[A] == k && (A & 1) == 0)
			{
				for (i = 0; i < node_EA; i++)
				{
```

```
                                        if (A & (int)pow(2, i))
                                                continue;
                                        mininum(i, A); //Vi가 A에 없을 경우 최적값 구하기
                                }
                        }
                }
        }
        mininum(0, max - 2);
        minlength = D[0][max - 2];
        return minlength;
}



void path(int i, int A)  //경로를 출력 하는 함수
{
        int j = P[i][A];
        printf("V%d -> ", i + 1);
        if (A - pow(2, j) != 0)
                path(j, (int)A - pow(2, j));
        else
                printf("V%d -> V%d\n", j + 1, 1);
}

void print(int min_lenth) //화면에 결과 출력 함수
{
        int i, j, a;

        if (min_lenth >= 1000)
                printf("불가능한 경로입니다.");
        else
        {
                printf("\n결과\n");
                printf("최소 경로 : ");                 //경로출력
                path(0, max - 2);
                printf("\n경로의 거리 : %d \n\n", min_lenth); //최소비용 출력
        }
}

int main()
{
        int i, j;       //for문에 사용되는 변수
        int tmp, middle_max, subset_count;   //부분집합 개수 구하기위해 필요한 변수
        int min_lenth;  //최단 길이 경로
```

```
printf("정점의 개수를 입력하십시오 : ");
scanf_s("%d", &node_EA);

max = (int)pow(2, node_EA);        //부분집합 최대 갯수

W = (int**)malloc(node_EA * sizeof(int));    //각 배열들 메모리 할당
D = (int**)malloc(node_EA * sizeof(int));
P = (int**)malloc(node_EA * sizeof(int));
subset = (int*)malloc(max * sizeof(int));

for (i = 0; i < node_EA; i++)
{
        D[i] = (int*)malloc(max * sizeof(int));
        P[i] = (int*)malloc(max * sizeof(int));
        for (j = 0; j < max; j++)
        {
                D[i][j] = P[i][j] = 0;     //배열 D,P 초기화
        }
}

printf("\n인접행렬을 입력하십시오.\n");
for (i = 0; i < node_EA; i++)
{
        W[i] = (int*)malloc(node_EA * sizeof(int));
        for (j = 0; j < node_EA; j++)
        { //파일로 부터 가중치 읽어서 배열 W에 저장
                scanf_s("%d", &W[i][j]);
        }
}


for (i = 0; i < max; i++)          //각 정점에서의 부분집합 갯수 구하기
{
        tmp = i;
        middle_max = max / 2;
        subset_count = 0;
        while (tmp > 0)
        {
                if (tmp - middle_max >= 0)
                {
                        tmp = tmp - middle_max;
                        subset_count++;
                }
                middle_max = middle_max / 2;
        }
```

```
            subset[i] = subset_count;
        }

        min_lenth = travel();//최소 경로 찾기


        print(min_lenth);    //결과 출력


        for (i = 0; i < node_EA; i++)      //각 배열들 메모리 할당 해제 하기
        {
            free(W[i]);
            free(D[i]);
            free(P[i]);
        }

        free(D);
        free(P);
        free(subset);
        free(W);
        return 0;
}
```

● 결과



● 해설

이 문제에서는 Dynamic programing을 구현하기 위해 집합을 사용하여야 하는데
집합을 어떻게 구현할지가 최대의 주안점이었습니다.

이를 해결하기 위해 비트마스크 방법을 활용하였습니다.
한 비트를 한 정점의 방문여부를 판단하는 flag로 설정하는 것으로 활용하였습니다.
예를 들어 문제에서처럼 5개의 정점 V1, V2, V3, V4, V5가 있을 때
이를 5자리의 비트(00000)로 표현하였습니다.
만약에 부분집합이 V1,V3,V5가 있을 경우 이를 비트로 표현하면
'10101' 로 나타낼 수 있습니다.

또한 부분집합 A에서 정점 하나를 빼는 경우에는
정점 자리수를 저장하는 s에 (int)pow(2,정점)을 넣어서
A-s로 정점 하나를 뺀 부분집합을 나타냈으며,

정점이 A에 있는지 없는지 궁금할 경우에는
A & s != 0을 통해 s번째 자리가 A에도 있으면 true값을 나타내는 형식으로 나타냈습니다.

Dynamic programing을 프로그래밍하여 컴퓨터로 시도해본 결과는
수기로 직접 풀었을 때와 결과가 같았으며
이를 통해 수기로 푼 결과가 정확하다는 것을 알 수 있었습니다.