**동적 계획법을 이용한 외판원 문제 해결**

알고리즘

숭실대학교 소프트웨어학부

20150213 김규호

신정훈 교수님

교재에 있는 Algorithm 3.11을 이용하여 외판원 문제를 해결하고 이를 검증한다.

목차

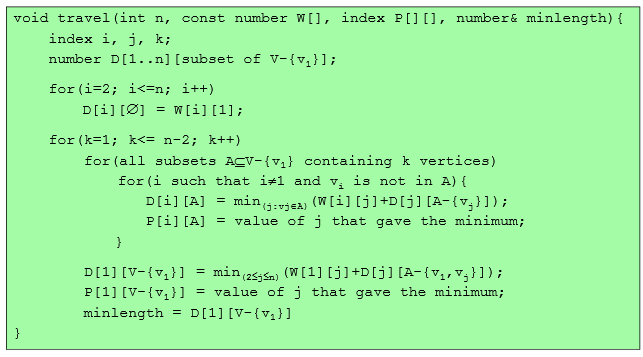
1. 수도코드

2. 그래프

3. 구현화면

4. 소스코드

1. 수도코드



2. 그래프

<그래프1> 1->3->4->2->1

1

3

1

9

2

8

6

7

6

<그래프2> 1->3->2->1

1

13

1

1

1

12

<그래프3> 1->2->3->1

1

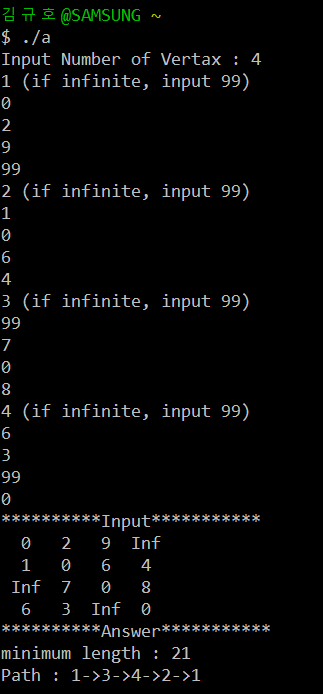
3

7

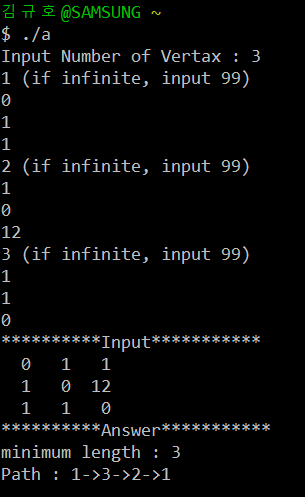
7

1

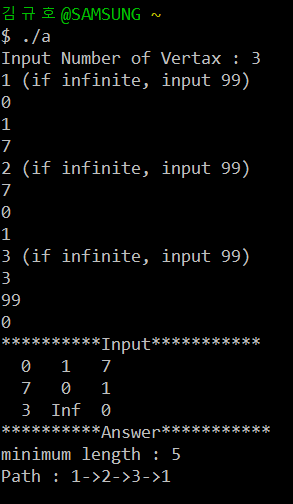
3. 구현화면



<그래프1> 1->3->4->2->1



<그래프2> 1->3->2->1



<그래프3> 1->2->3->1

4. 소스코드

#include<stdio.h>

/\*

\* printf(), scanf(), for, if, continue, switch

\*/

#include<stdlib.h>

/\*

\* calloc()

\*/

int \*IN;//Change index to bit(ex. 2->1, 3->10, 4->100, ...)

int V;//Set of all exclude vertax1

int minimum(int a, int b)

{

return a < b ? a : b;//return smaller thing

}

int setsize(int A)//check size of set

{

int result;

int tmp = A;

for(result = 0;tmp > 0;tmp = tmp>>1)

if(tmp & 1)

result++;//if A has bit, result + 1

return result;

}

void travel(int n, int \*\*W, int \*\*P, int \*minlength)

{

int i, j, k;

int check;

int A;//A : subset of V

int \*\*D;//D : all weight(Vertax -> {})

//make D[n][V]

D = (int \*\*) calloc(n \* (V + 1), sizeof(int \*));

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

\*(D + i) = calloc(V + 1, sizeof(int));

//D[i][Empty Set] = W[i][{1}]

for(i = 1; i < n; i++)

D[i][0] = W[i][0];

for(k = 1; k <= n - 2; k++)

for(A = 1; A <= V; A++)

if(k == setsize(A))//check that size of set is k

for(i = 1; i < n; i++)

{

if(A & IN[i])

continue;

else

{

D[i][A] = 99;

check = D[i][A];

for(j = 1; j < n; j++)

{

if(A & IN[j])

{

D[i][A] = minimum(D[i][A], W[i][j] + D[j][A ^ IN[j]]);

if(check != D[i][A])

{

P[i][A] = j + 1;

check = D[i][A];

}

}

}

}

}

D[0][V] = 99;

check = D[0][V];

for(j = 1; j < n; j++)

{

D[0][V] = minimum(D[0][V], W[0][j] + D[j][V - IN[j]]);

if(check != D[0][V])

{

P[0][V] = j + 1;

check = D[0][V];

}

}

\*minlength = D[0][V];

}

void PrintPath(int \*\*P, int from, int A)

{

int tmp = P[from][A];

if(tmp ==0)

{

printf("1\n");

return;

}

else

{

printf("%d->", tmp);

PrintPath(P, tmp-1, A - IN[tmp-1]);

}

}

int main()

{

int i, j;

int \*\*W;//W : first weight(Vertax -> 1)

int \*\*P;//Path array

int N;//Size of input(Number of Vertax)

int minlength;//return value of travel()

printf("Input Number of Vertax : ");

scanf("%d", &N);

//make V value

V = (1<<N-1) - 1;

//make IN[N]

IN = (int \*) calloc(N, sizeof(int));

for(i = 1; i < N; i++)//IN[0] : vertax1 = 0

IN[i] = 1<<i-1;

//make P[N][V]

P = (int \*\*) calloc(N \* (V + 1), sizeof(int \*));

for(i = 0; i < N; i++)

\*(P + i) = calloc(V + 1, sizeof(int));

//make W[N][N]

W = (int \*\*) calloc(N \* N, sizeof(int \*));

for(i = 0; i < N; i++)

\*(W + i) = (int \*) calloc(N, sizeof(int));

//Get W[N][N]

for(i = 0; i < N; i++)

{

printf("%d (if infinite, input 99)\n", i + 1);

for(j = 0; j < N; j++)

scanf("%d", &W[i][j]);

}

printf("\*\*\*\*\*\*\*\*\*\*Input\*\*\*\*\*\*\*\*\*\*\*\n");

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

for(j = 0; j <N; j++)

if(W[i][j] == 99)

printf(" Inf");

else

printf(" %2d ", W[i][j]);

printf("\n");

}

printf("\*\*\*\*\*\*\*\*\*\*Answer\*\*\*\*\*\*\*\*\*\*\*\n");

travel(N, W, P, &minlength);

printf("minimum length : %d\n", minlength);

printf("Path : 1->");

PrintPath(P, 0, V);

printf("\n");

return 0;

}