韓國數學教育學會誌<數學教育>
1985. 6, Vol. XXIII, No. 2

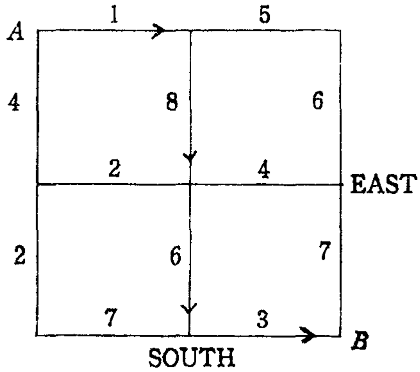
Journal of the Korea Society of Mathematical Education
Jun. 1985, Vol. XXIII, No. 2

Bellman의 最適原理를 위한 Dynamic Programming

江原大學校 白 淸 鎬

1. 緖

〈그림 1〉과 같은 2×2 格子문제를 생각해보자



〈그림 1〉 A에서 B로 가는 최소 시간은 얼마인가?

이 문제에서 각 구간에 쓰인 수는 그 구간을 통과하는 데 필요한 시간(分)이라고 하자. 그 시간은 구간에 따라 다르다. 이 때 A에서 B로 가는 최소 시간은 얼마인가? 단, 각 格子點에서의 진행방향은 동쪽(E) 아니면 남쪽(S)뿐이다.

진행방향을 동쪽은 E, 남쪽은 S로 나타내기로 하면 〈그림 1〉에서의 화살표로 표시된 경로는 *ESSE*이다. A에서 B로 가는 가능한 모든 경로는 6가지이며 그 경로와 그에 대한 소요시간은 〈표 1〉에 나타내었다.

〈표 1〉 〈그림 1〉의 A에서 B까지의 모든 경로와 시간

경 로	시 간(分)
ESSE	1+6+8+3=18
EESS	1+5+7+6=19
ESES	1+6+4+6=17
SEES	2+2+4+6=14
SSEE	2+4+7+3=16
SESE	2+2+8+3=15

따라서 최소 시간은 14분이며 최적 경로는 *SEES*이다. 이 문제를 풀기 위해 대부분 각 격자점에서 최소 시간으로 찾아가려 하지만 (예를 들면 〈그림 1〉에서 *EESS*로 시간은 19분) 이것은 잘못된 생각이다. 이 방법으로는 최적 경로를 찾을 수 없다.

이러한 문제를 해결하는 데 필요한 연산 횟수를 계산해보자. 우선 순열과 조합의 원리를 응용하면, 가능한 경로의 수는 E, E, S, S를 1열로 세우는 방법의 수와 같으므로 $4!/(2!2!)=6$가지이며 각 경로마다 3번의 덧셈이 필요하므로 (경로를 나타내는 글자수 보다 하나 적은 횟수만큼의 덧셈) 모두 $6\times3=18$회의 덧셈이 필요하다.

〈표 1〉에서 최소 시간을 찾기 위하여
18과 19를 비교하여 18이 작고
18과 17을 비교하여 17이 작고
17과 14를 비교하여 14가 작고
14와 16을 비교하여 14가 작고
14와 15를 비교하여 14가 작다.

따라서 A에서 B에 이르는 최소 시간은 14분이다. 여기에는 5회의 비교(경로의 수보다 하나 적다)가 필요하다. 그래서 연산 횟수는 〈표 2〉와 같다.

〈표 2〉 2×2 격자문제의 연산 횟수

	연산 횟수
덧 셈	18
비 교	5
계	23

이제 10×10격자문제를 생각해보자.

앞의 방법으로 문제를 풀려면 대략 얼마나 걸릴 것인가? 연산 횟수를 계산해보면 모든 경로

의 수는 20!/(10!10!)=184,756, 각 경로마다 19회의 덧셈이 필요하면 184,755회의 비교가 필요하다.

〈표 3〉 10×10 격자문제의 연산 횟수

	연산 횟수
덧 셈	3,510,364
비 교	184,755
계	3,695,119

이 문제를 식사시간, 수면시간 등을 포함하여 푼다면 한 사람이 최소한 45일이 걸린다. 1초에 100,000번의 연산을 할 수 있는 computer로는 37초 정도 걸린다.

2. Computer의 制限性

30×30 격자문제의 경우를 생각해보자. 이러한 문제는 大都市의 경우에 흔히 나올 수 있는 문제이다. 이 경우 경로의 수는 60!/(30!30!)≒1.18×10^{17}이며 각 경로마다 59회의 덧셈이 수행되어야 한다. 그리고 나서 비교가 이루어진다. 분명히 연산 횟수는 10^{18}을 넘게 되며 초당 100,000번의 연산을 할 수 있는 computer로서도 1年에 3.1536×10^{12}회의 연산을 할 수 있을 뿐이며, 따라서 이러한 computer를 사용한다 해도

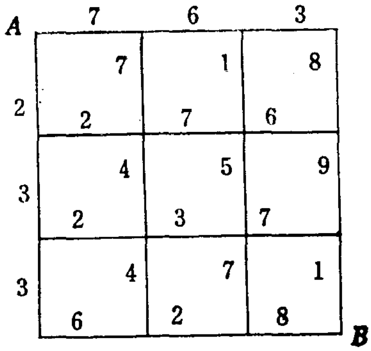
$$\frac{10^{18}}{3.1536\times10^{12}}>\frac{10^{18}}{10^{13}}=10^5=100,000$$

즉, 100,000年 이상이 걸린다. 이러한 점은 어떤 문제를 해결하고자 하는 computer algorithm을 작성할 때 연산 횟수에 대한 점을 고려해야 한다는 것을 말해주고 있다.

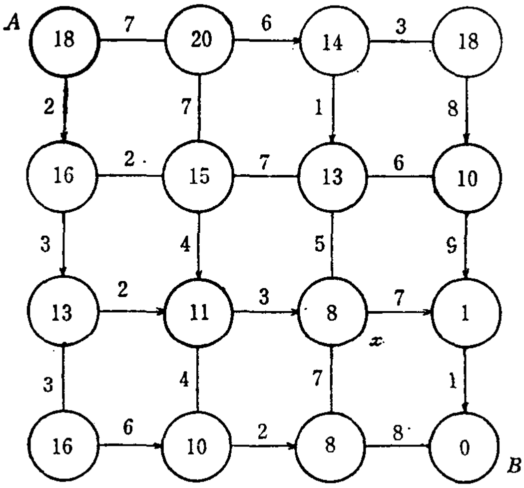
3. Dynamic programming

〈그림 2〉와 같은 3×3 격자문제를 보기로 삼아 Dynamic programming의 例示를 하고자 한다.

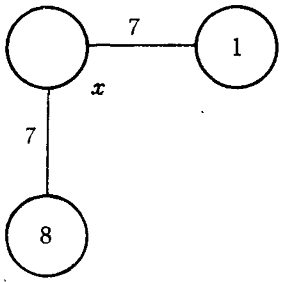
이 방법은 B에서 출발하여 北, 西의 방향으로 順次的으로 後進하면서 각 격자점에서 "이곳에서 B까지의 최소 시간과 방향은 무엇인가?"하고 생각해본다. 이 결과를 〈그림 3〉에 나타내었다.



〈그림 2〉 3×3 격자문제



〈그림 3〉 각 격자점에서 B에 이르는 최소 시간



〈그림 4〉 격자점 x의 최소 시간 결정

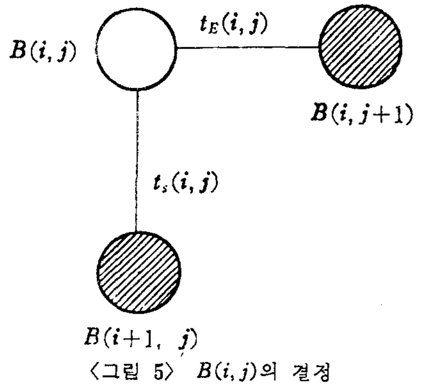
〈그림 4〉에 따로 그린 격자점 x에서의 최소 시간 결정에 대해서 알아보자. 만일 x에서 동쪽으로 가면 B에 이르는 시간은 7+1=8분이며 남쪽으로 가면 7+8=15분이다. 8이 작으므로 원 안에 8을 쓰고 화살표는 동쪽으로 그린다. 원 안에 쓰인 수는 바로 그 격자점에서 B까지의

소 시간을 의미한다. 이렇게 하면 각 격자점에 는 3번 이하의 연산으로 충분하다. (가장자리 있는 격자점에 대해서는 연산 횟수가 1회이). 〈그림 3〉에서 A에서 B에 이르는 최소 시 은 18분이며 適正 經路는 화살표를 따라 SSE-ES이다.

Dynamic programming에서 격자의 크기는 각 자점에서 필요한 3회 이하의 연산 횟수에 영 을 미치지 않는다. 따라서 30×30 격자문제에 는 31×31=961개의 격자점이 있으므로 3,000 이하의 연산으로 이 문제를 해결할 수 있으 초당 100,000회의 연산을 하는 computer로는 100초 이하로 충분하다. 앞의 방법으로는 같 computer로 100,000년 이상 걸리던 것과 비 하면 Dynamic programming의 위력을 실감할 있을 것이다. Bellman의 最適 理論을 이용한 격자점에서 B에 이르는 최소 시간을 구하는 namic programming의 방법은 다음과 같다.

이 문제를 computer로 해결하기 위해 각 격자 들을 元素로 하는 行列 B를 만든다. 즉, A점 $B(1,1)$으로 B점을 $B(4,4)$로 하는 4次의 正 행렬 B를 만들어 〈그림 3〉에서 제 i 행 제 j 열 자점의 圓 내부의 값이 행렬 B의 제 i 행 제 j 열 소 $B(i,j)$의 값이 되도록 하는 함수를 만들면 다. 즉, 격자점 (i,j)에서 함수 $B(i,j)$가 격자 (i,j)에서 목적지 B〔즉 $B(4,4)$〕에 이르는 최 시간을 나타내도록 하면 된다. 함수값이 결 되는 상황은 〈그림 5〉에 나타내었다.

여기서 $B(i, j+1)$ 및 $B(i+1, j)$의 값은 이미 고 있는 것이다. $t_S(i,j)$는 격자점(i,j)에서 남



〈그림 5〉 $B(i,j)$의 결정

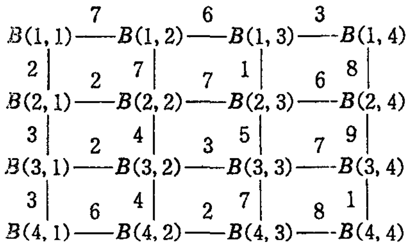
쪽으로 한 구간 가는 데 걸리는 시간을 나타내며 $t_E(i,j)$는 동쪽으로 가는데 필요한 시간을 나타낸다. 그래서

$$B(i,j)=\min\{B(i,j+1)+t_E(i,j),\ B(i+1,j)+t_S(i,j)\}$$

로 함수 $B(i,j)$를 정의한다.

4. 풀이의 實際 및 Programming

예를 들어 〈그림 2〉의 문제를 푸는 과정을 상세히 설명하면 다음과 같다.



〈그림 6〉 격자문제와 행렬 B

〈그림 6〉을 computer에 이용하기 위한 행렬 TS 및 TE는 다음과 같다.

$$TS=\begin{pmatrix}2&7&1&8\\3&4&5&9\\3&4&7&1\end{pmatrix}\qquad TE=\begin{pmatrix}7&6&3\\2&7&6\\2&3&7\\6&2&8\end{pmatrix}$$

계산은 逐次的으로 진행되며 그 과정 및 결과는 다음과 같다.

$$B(4,4)=0$$
$$B(3,4)=B(4,4)+TS(3,4)=1$$
$$B(4,3)=B(4,4)+TE(4,3)=8$$
$$B(2,4)=B(3,4)+TS(2,4)=10$$
$$B(3,3)=\min\{B(3,4)+TE(3,3),\ B(4,3)+TS(3,3)\}=8$$
$$B(4,2)=B(4,3)+TE(4,2)=10$$
$$B(1,4)=B(2,4)+TS(1,4)=18$$
$$B(2,3)=\min\{B(2,4)+TE(2,3),\ B(3,3)+TS(2,3)\}=13$$
$$B(3,2)=\min\{B(3,3)+TE(3,2),\ B(4,2)+TS(3,2)\}=11$$
$$B(4,1)=B(4,2)+TE(4,1)=16$$

$$B(1,3)=\min\{B(1,4)+TE(1,3),\; B(2,3)+TS(1,3)\}=14$$
$$B(2,2)=\min\{B(2,3)+TE(2,2),\; B(3,2)+TS(2,2)\}=15$$
$$B(3,1)=\min\{B(3,2)+TE(3,1),\; B(4,1)+TS(3,1)\}=13$$
$$B(1,2)=\min\{B(1,3)+TE(1,2),\; B(2,2)+TS(1,2)\}=20$$
$$B(2,1)=\min\{B(2,2)+TE(2,1),\; B(3,1)+TS(2,1)\}=16$$
$$B(1,1)=\min\{B(1,2)+TE(1,1),\; B(2,1)+TS(1,1)\}=18$$

따라서 A에서 B에 이르는 최소 시간은 $B(1,1)=18$이며 $B(i,j)$의 값은 격자점 (i,j)에서 목적지 B에 이르는 최소 시간이다. 最適 方向을 추적해보면,

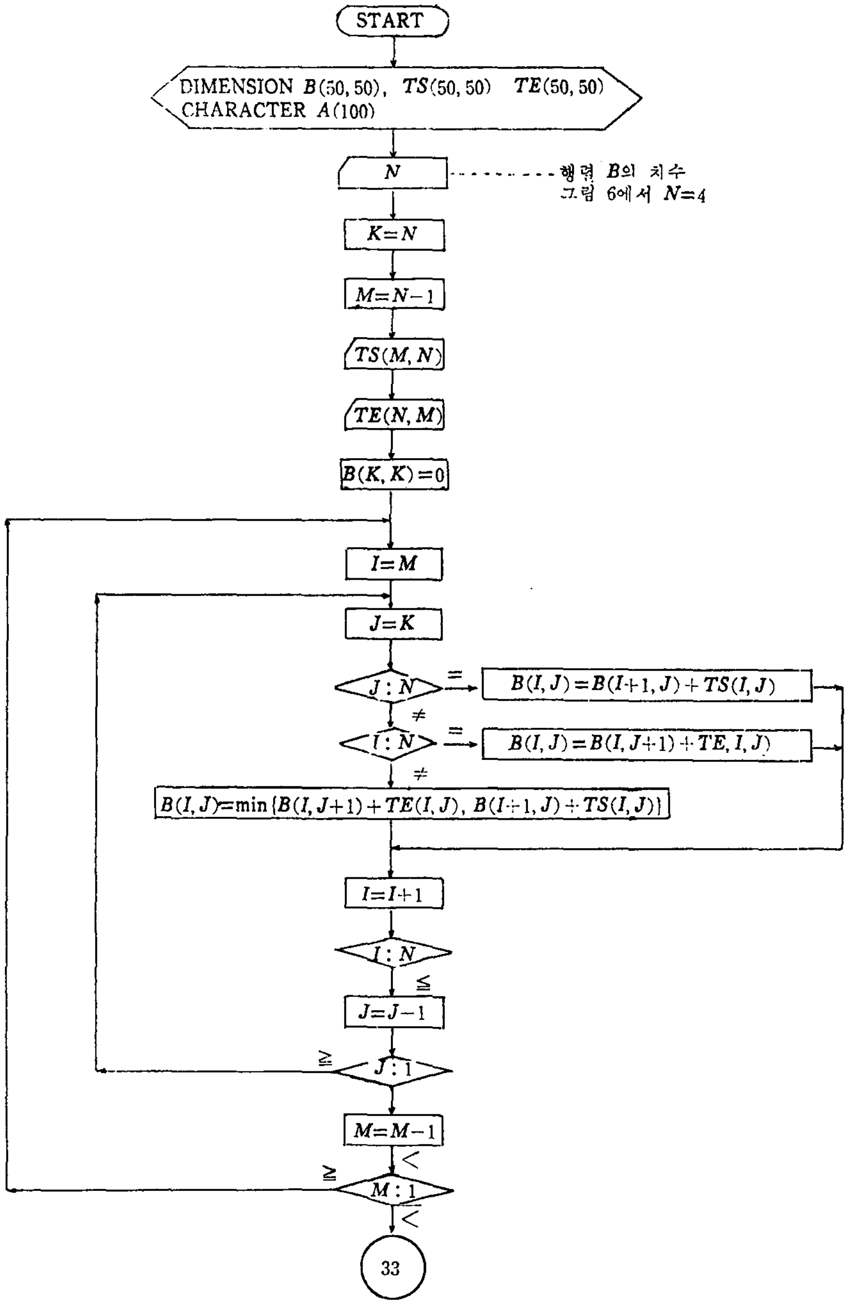
방향 : $B(1,1)\underset{S}{\rightarrow}B(2,1)\underset{S}{\rightarrow}B(3,1)\underset{E}{\rightarrow}B(3,2)\underset{E}{\rightarrow}$
$B(3,3)\underset{E}{\rightarrow}B(3,4)\underset{S}{\rightarrow}B(4,4)$

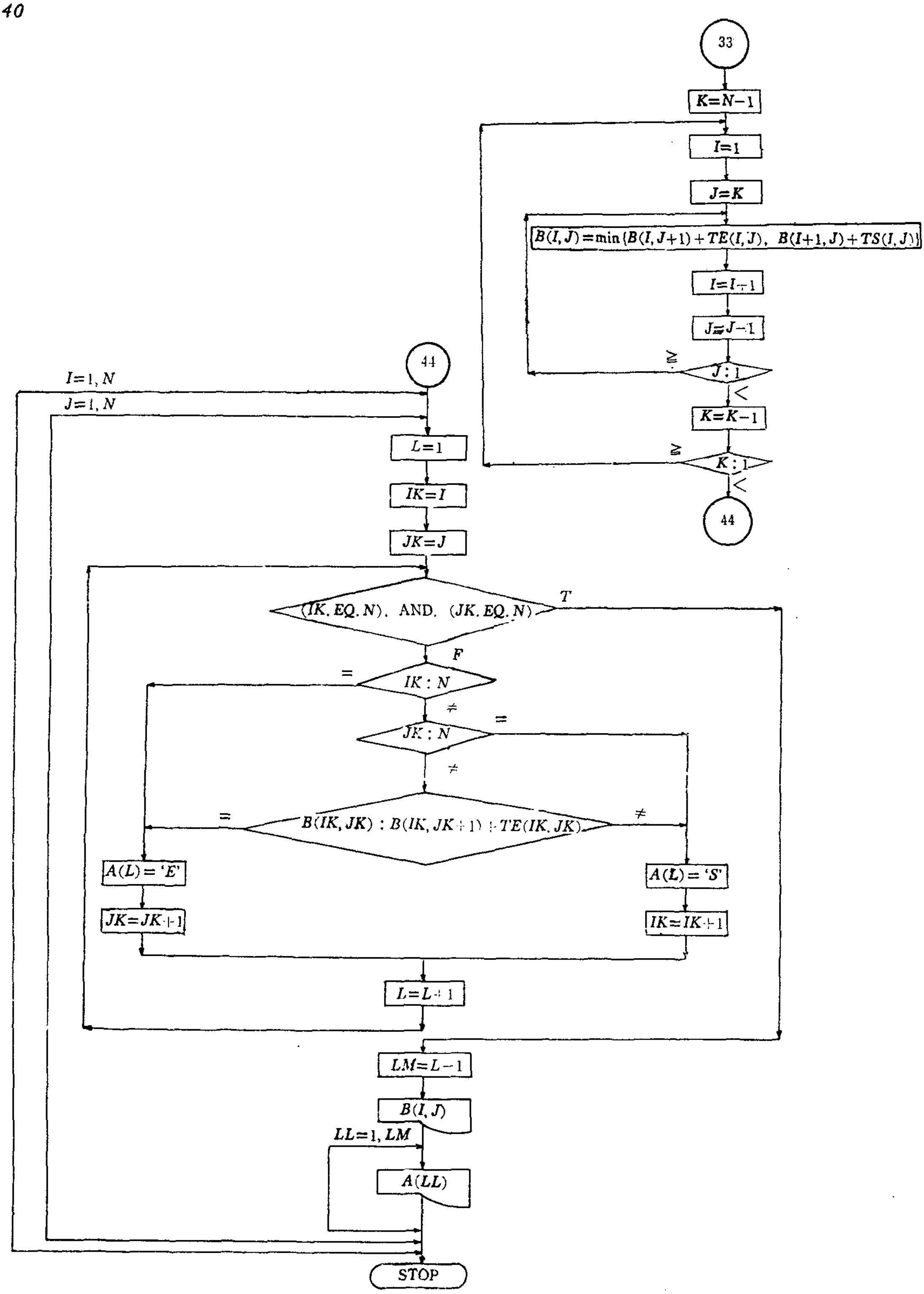
따라서 A에서 B에 이르는 최적 방향은 $SSEEES$이다.

지금까지의 문제를 풀기 위한 Flow-chart는 다음과 같다.

참 고 문 헌

1. George L. Nemhauser, *Introduction to Dynamic Programming*, John Wiley and Sons, Inc., New York, 1966.


START
DIMENSION B(50, 50), TS(50, 50) TE(50, 50)
CHARACTER A(100)
N
행렬 B의 치수
그림 6에서 N=4
K=N
M=N−1
TS(M, N)
TE(N, M)
B(K, K)=0
I=M
J=K
J : N
=
B(I, J)=B(I+1, J)+TS(I, J)
≠
I : N
=
B(I, J)=B(I, J+1)+TE, I, J)
≠
B(I, J)=min{B(I, J+1)+TE(I, J), B(I+1, J)+TS(I, J)}
I=I+1
I : N
≦
J=J−1
≧
J : 1
M=M−1
<
≧
M : 1
<
33



33
K=N−1
I=1
J=K
B(I, J)=min{B(I, J+1)+TE(I, J), B(I+1, J)+TS(I, J)}
I=I+1
J=J−1
≧
J : 1
<
K=K−1
≧
K : 1
<
44
44
I=1, N
J=1, N
L=1
IK=I
JK=J
(IK. EQ. N). AND. (JK. EQ. N)
T
F
IK : N
=
≠
JK : N
=
≠
B(IK, JK) : B(IK, JK+1)+TE(IK, JK)
=
≠
A(L)='E'
JK=JK+1
A(L)='S'
IK=IK+1
L=L+1
LM=L−1
B(I, J)
LL=1, LM
A(LL)
STOP