Bridge and Torch Problem Using Dynamic Programming

Varian Caesar | 13514041[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganeca 10 Bandung 40132, Indonesia
[1]13514041@std.stei.itb.ac.id

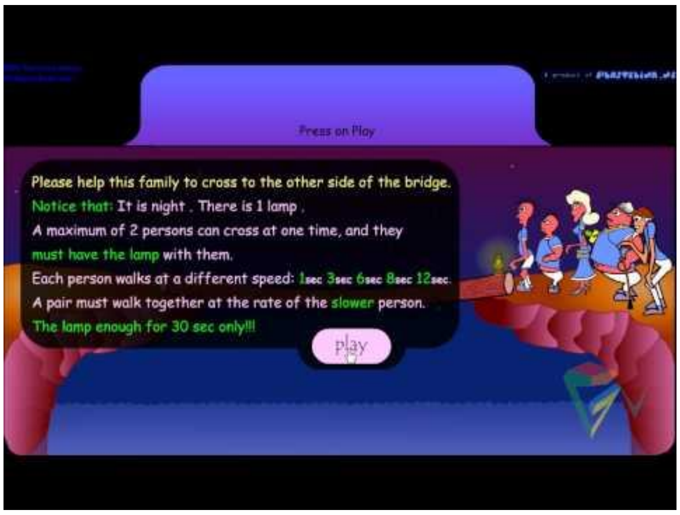
***Abstract*—Bridge and Torch Problem is a popular Computer Science (CS) problem created by Richard Hovasse. It is a simple problem but very interesting and quite difficult to answer with *brute force*. The problem deals with some people crossing the dark and long bridge, the goal is to move all people from one side to another side with some constraints and minimum time. There are many way to solve this problem, but in this paper the author will show how to deal with this problem using Dynamic Programming. The idea of Dynamic Programming is to take the optimal solution for each step that become the subset of real solution.**

***Keywords*— Bridge and Torch Problem, Dynamic Programming, optimal solution**

I. Introduction

Nowadays, game is very popular among us. There are so many genres of the game, including logic game. We needs to think hard to solve this logic game or we can do a brute force attack to them, but it will be very exhausting. One of most popular logic game is Bridge and Torch Problem that asked by Richard Hovasse. This logic game seems simple but contains interesting puzzle in it and we can't underestimate the logic behind it.

Bridge and Torch problem start with a group of person where persons > 2 and they needs to move from one side of bridge to another side by crossing the long and dark bridge. Unfortunately, the bridge can only hold maximum of C person on it. They have a torch to light up the bridge but the batteries is running out in just a few minutes, and the light from it already reduced. From all people you know that the pace of walking is different from people-1 to people-k. If two or more people, where people < C travel from one side to another, because the torch's light is not too bright, the travel from one side to another must follow the slower person's pace of walk and since the bridge is long you can't throw or roll the torch from another side back to the initial side. The only way to return the torch is send a person back from another side to initial side, bring the torch back by themselves.



Please help this family to cross to the other side of the bridge.
Notice that: It is night . There is 1 lamp .
A maximum of 2 persons can cross at one time, and they must have the lamp with them.
Each person walks at a different speed: 1sec 3sec 6sec 8sec 12sec.
A pair must walk together at the rate of the slower person.
The lamp enough for 30 sec only!!!

Figure 1 Bridge and Torch Problem

As the problem said before, the torch is running out of batteries, so it will last for a few minutes. You must help these person crossing the bridge with the optimal time so all of them can cross the bridge before the torch is completely turned off.

Generally, this problem can be solved by model it into graph and we search the minimum cost for each vertex using BFS, DFS, UCS, etc. But in this paper the author wants to show us how to solve this problem by using the Dynamic Programming.

II. Basic Theories

A. *Dynamic Programming*

Dynamic Programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems in recursive manner, solving each those subproblems just once and storing their solutions using data structure. The next time the same subproblems occurs, instead of recomputing its solution, one simply looks up the previously computed solution.

Dynamic Programming algorithms are often used for optimization. A dynamic programming algorithm will examine the previously solved subproblems and will combine their solution to give the best solution for the given problem. **In comparison**, a greedy algorithm treats the solution as some sequence of steps and pick the locally optimal choice at each step.Using a greedy algorithm is less optimal because picking locally optimal choices may result in bad global solution.

The Characteristics of Dynamic Programming :

1. Problems can be divided into some subproblems, for each step we take one solution
2. For each step there are some state that related to it. In general, state is all possibilities of input at that step
3. Result from each step will transformed from the corresponding state to next state in the next step
4. The cost from one step will increase steadily with the increasing step
5. The cost in a step determined by the cost from previous step and the cost in that step
6. Best Solution in one step is independent from the other solution
7. Recursive manner, the best solution in stage k will give the best solution in stage k+1

There are two approach of Dynamic Programming : Up-down and bottom-up. If $x_1,x_2,x_3,\ldots x_n$ is state variable for step 1,2,3,..., n then :

1. Up-down : Dynamic program start at stage 1 to stage 2, 3 and so on until stage n. The order are $x_1,x_2,x_3,\ldots x_n$.
2. Bottom-up : Dynamic program start at stage n and moving backward to stage n-1, n-2 and so on until stage 1. The order are x_n, x_{n-1}, …, x_1

Sample problem that can be solved using Dynamic Programming :

1. Shortest Path
2. Capital Budgeting
3. Integer (1/0) Knapsack
4. TSP

III. Bridge and Torch Using Dynamic Programming

For the application of Dynamic programming in this problem, we create a simple Bridge and Torch Problem with 4 people that must cross from southern bank to northern bank using the bridge and a torch (because its at night). The bridge is long and dark so you cannot pass the torch by throwing or roll it in the ground. The person, says A, B, C and D walk with different pace as follows :

Person	Time to travel to another side
A	1
B	3
C	8
D	10

Table 1 Time to travel for each person

Unfortunately the bridge was built in 1890 so it’s very old and maximum only 2 people that can cross the bridge at the same time. The torch will running out of batteries in just a few minutes, can you help them cross the bridge with the optimum time ?

There are 2 approach of Dynamic Programming for this problem :

A. Dynamic programming using Process table

The conceptual framework we use to construct a mathematical model for the problem above is sequential decision processes. That is, we regard the problem under consideration as a sequential decision problem. We assume that there will be k crossings, j = 1, 2, …, k.

Crossing j < k consist of two parts : a group is moved from the southern bank to the northern banks and then one person returns the torch from the northern bank to the southern bank. The last crossing, j = k, consist only of the first part, as there is no need to return the torch.

Decision variables :
Let

X_j = group of persons moving from south to north in the j-th crossing, j = 1, 2, …, k

Y_j = person returning the torch to the southern bank after the j-th crossing, j = 1, 2, …, k

State Variables :
Let

S_j = group of person on the southern bank just before the j-th crossing, j = 1, 2, …, k

State transition :
Given the above definitions, it follows that the dynamics of the state variables is governed by the following transition function :

$S_{j+1} = (S_j - X_j) \cup Y_j$, j = 1, 2, …, k

Objective function :
Let

$T(X_j, Y_j) = t(X_j) + t(Y_j)$

Then by definition, $T(X_j, Y_j)$ is the time to complete a travel from southern bank to northern bank including the return of the torch to the souther bank, So it follows :

$$F(X,Y) = T(X_1, Y_1) + \ldots + T(X_{k-1}, Y_{k-1}) + t(X_k)$$

The duration of the last crossing is equal to $t(X_k)$ as there is no need to return the torch to the southern bank.

The person who returns the torch from northern bank to the southern bank must be the one with the fastest time. That is, given that group S_j is currently on the northern bank, person p will return the torch to the southern bank where t(p) = min {t(i) , where i in S_j }. Let $p(S_j, X_j)$ denote this person, then :

$S_{j+1} = Next(S_j, X_j) = (S_j – X_j) U p(S_j, X_j)$

Let $D(S_j)$ denote the set of feasible values of X_j. It follows then that we can set:

$D(S_j) = \{S_j\}$
$D(S_j)$ = {G : G is a subset of S_j such that 2 <= |G| <= C.
If |G| < C then t(G) < t(p) for all p in S_j}, |S| > C,
C is the capacity of the bridge.

We now derive a dynamic programming functional equation for this problem. Solving this equation will yield an optimal policy for the problem. To accomplish this, define the state of the process to be the group of persons on the southern bank, namely G.
Let :

F(S) = minimal crossing time required to move group S from south to north

With this in mind, let

F(S,G) = minimal crossing time required to move group S from south to north, given that subgroup G is moved first

Lemma 1 :
F(S) = t(S), |S| <= 2
F(S) = min {T(S,G) + F(Next(S,G))

Finally let D*(S) denote the set of optimal values of G associated with F(S) :

D*(S) = {S}, 1 <= |S| <= 2

D*(S) = {G* in D(S): T(S,g*) + F(Next(S,G*)) = min

{T(S,G) + F(Next(S,G)): G in D(S) }, |S| > 2

Example

Consider the example given at the beginning of this chapters:

j	A	B	C	D
t(j)	1	3	8	10

Table 2

Step 1

S	F(S)	D*(S)
{A}	1	{{A}}
{B}	3	{{B}}
{C}	8	{{C}}
{D}	10	{{D}}
{A,B}	3	{{A,B}}
{A,C}	8	{{A,C}}
{A,D}	10	{{A,D}}
{B,C}	8	{{B,C}}
{B,D}	10	{{B,D}}
{C,D}	10	{{C,D}}

Table 3 Result of Step 1

Step 2
We can now solve F(S) for all S such that |S| = 2 + 1 = 3. As follows :
S = { {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D} }

For S = {A,B,C} we construct table as follows :

G	{A,B}	{A,C}	{B,C}
p(S,X)	A	A	B
Next(S,G)	{A,C}	{A,B}	{A,B}
t(G)	3	8	8
t(p(S,G))	1	1	3
F(Next(S,G))	8	3	3
F(S,G)	12	12	14

Table 4 F({A,B,C})

It follows that F({A,B,C}) = min {12,12,14} = 12. There are two optimal decision for this subproblem, namely G = {A,B} and G = {A,C}, hence D*({A,B,C}) = {{A,B},{A,C}}

Similiarly, we create the table for the rest of S

For S = {A,B,D} we construct table as follows :

G	{A,B}	{A,D}	{B,D}
p(S,X)	A	A	B
Next(S,G)	{A,D}	{A,B}	{A,B}
t(G)	3	10	10
t(p(S,G))	1	1	3
F(Next(S,G))	10	3	3
F(S,G)	14	14	16

Table 5 F({A,B,D})

It follows that F({A,B,D}) = min {14,14,16} = 14. There are two optimal decision for this subproblem, namely G = {A,B} and G = {A,D}, hence D*({A,B,D}) = {{A,B},{A,D}}

For S = {A,C,D} we construct table as follows :

G	{A,C}	{A,D}	{C,D}
p(S,X)	A	A	B
Next(S,G)	{A,D}	{A,C}	{A,B}
t(G)	8	10	10
t(p(S,G))	1	1	3
F(Next(S,G))	10	8	3
F(S,G)	19	19	16

Table 6 F({A,C,D})

It follows that F({A,C,D}) = min {19,19,16} = 16. There are only one optimal decision for this subproblem, namely G = {C,D}, hence D*({A,C,D}) = {{C,D}}

For S = {B,C,D} we construct table as follows :

G	{B,C}	{B,D}	{C,D}
p(S,X)	A	A	A
Next(S,G)	{A,D}	{A,C}	{A,B}
t(G)	8	10	10
t(p(S,G))	1	1	1
F(Next(S,G))	10	8	3
F(S,G)	19	19	14

Table 7 F({B,C,D})

It follows that F({B,C,D}) = min {19,19,14} = 14. There are only one optimal decision for this subproblem, namely G = {C,D}, hence D*({B,C,D}) = {{C,D}}

Finally we can construct the table for step 2 as follows :

S	F(S)	D*(S)
{A,B,C}	12	{{A,B}, {A,C}}
{A,B,D}	14	{{A,B}, {A,D}}
{A,C,D}	16	{{C,D}}
{B,C,D}	14	{{C,D}}

Table 8 Solution Table for step 2

Step 3
Now we can deal with the group whose cardinality is 4. There is only one such group, namely S = P = {A,B,C,D}. For this case, the set of feasible decisions is as follows :

D({A,B,C,D}) = {{A,B}, {A,C}, {A,D}, {B,C}, {B,D}, {C,D}}

The following table explains how the value of F({A,B,C,D}) is computed :

G	{A,B}	{A,C}	{A,D}	{B,C}	{B,D}	{C,D}
p(S,G)	A	A	A	B	B	C
Next(S,G)	{A,C,D}	{A,B,D}	{A,B,C}	{A,B,D}	{A,B,C}	{A,B,C}
t(G)	3	8	10	8	10	10
t(p(S,G))	1	1	1	3	3	8
F(Next(S,G))	16	14	12	14	12	12
F(S,G)	20	23	23	25	25	30

Table 9 Solution

Thus, F(P) = F({A,B,C,D}) = 20 and the optimal decision is G = {A,B}, hence D*({A,B,C,D}) = {{A,B}}. This means that the next state of the process will be Next({A,B,C,D}, {A,B}) = {A,C,D}. The optimal decision at this state is G = {C,D}. This will change the satate of the process to Next({A,C,D}, {C,D}) = {A,B}. The optimal solution for this state is G' = {A,B}. The state resulting from this decision will then be E = {}. It follows then that the optimal policy is p = {{A,B},{A},{C,D},{B},{A,B}} or :
A and B move from south to north
A come back to south
C and D move from south to north
B come back to south
A and B move from south to north

B. Dynamic programming Shortest Path reduction

We already know how to solve Bridge and torch problem using dynamic programming with decision table. There is one more way to solve this problem using Dynamic programming, its by reducing the problem into Shortest path problem using graph.

Example

With the same example as before, we can represent the problem using graph. Each node represent a step and value on the edge represent the time needed to do a travel and come back again to the south. Value on the node represent the person on the northern bank. So the graph is as follows :

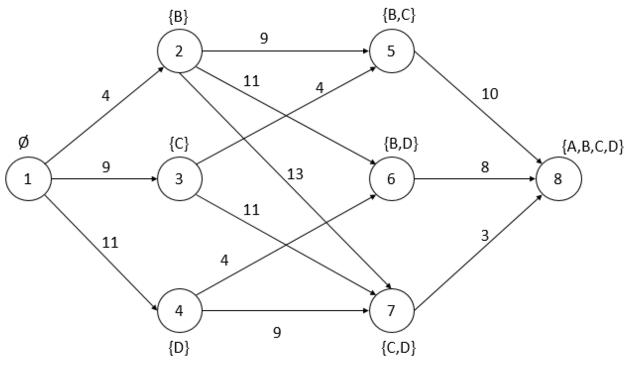


Figure 2 Bridge and Torch's graph

Now our task is to find the minimum cost from node 1 to node 8 that represent the solution of Bridge and torch problem.

Decision Variables :
Let

X_k = node that we must take at step k
$Cs,_{xk}$ = cost from s to X_k

State variables :
Let

s = set of node that we can take at step k + 1

State Transition :
Given the above definitions, it follows that the dynamics of the state variables is governed by the following transition function :

$f_k(X_k, s) = C_{xk,s} + f_{k-1}(X_k)$

Objective function :
This recursive show the shortest path from s to X_4 at step k:

$f_1(s) = C_{xk,s}$

$f_k(s) = \min \{ C_{xk,s} + f_{k-1}(X_k) \}, k = 2,3,4$

So now we can start to solve this problem using Dynamic Programming :

Step 1

s	$f_1(s)$	X_1
2	4	1
3	9	1
4	11	1

Table 10 Basis for Shortest Path

Because this is a basis, X_1 is always the start node, not to mention node 1.

Step 2

s \ X_2	2	3	4	$f_2(X_2)$	X_2
5	13	13	-	13	2 or 3
6	15	-	15	15	2 or 4
7	17	20	20	17	2

Table 11 Table for Step 2

As you can see there exist value “-“ it means there’s no edge connecting both node, you can also give it a value of infinity to ensure that edge never been selected.

Step 3

s \ X_3	5	6	7	$f_3(X_3)$	X_3
8	23	23	20	20	7

Table 12 Table for Step 3

Now we must reconstructing the global solution using the table we get so far. We know that the minimal $f_3(X_3)$ = 20 and it implies the X_3 must be node 7, the solution so far = {7-8}. Now we back to step 2, at s = 7 we get the value of f_2 (7) = min { $C_{2,7}$ + $f_1(X_1)$ } = 17 and it implies the X_2 must be node 2. The solution so far = {2 – 7 – 8}. Now we look at the step 1, step 1 is a basis so the X_1 must be the start node, not to mention node 1. The solution = {1-2-7-8}.

Now we already know the node transition, our final task is represent that node into the step of travel for each person :

From node 1 to node 2, A and B move to the north and then A come back to the south = 4 minute

From node 2 to node 7, C and D move to the north and then B come back to the south = 13 minute

From node 7 to node 8, A and B move to the north = 3 minute.

Total = 20 minute.

IV. Analysis and Implementation

Using Dynamic Programming to determine the step in Bridge and Torch problem always give an optimal solution. Because the subproblems of Bridge and Torch problem is always be the subset of the global solution, so solve the subprblems will solve the entire problems too.

A. Analysis

Before we do the implementation, we must know what exactly Dynamic Programming do to solve this problem. As we can see, Dynamic Programming is different with the greedy. Greedy algorithm only take decision based on the local optima, the difference between Dynamic Programming and Greedy might been seen on step 2 from the example before, as person C and D are both the slowest person on the group we can save so much time by traveling them together. But with greedy, we don’t ever considering the time passed, so we pair each C and D with A to reach local optima. This decision of greedy algorithm will waste the speed of person A as the fastest person on the group.

This problem also can be solved using graph algorithm like DFS, BFS, UCS, and A-Star. But solving using these algorithm will need some space in your memory because they need to expand the tree and manipulating pointer.

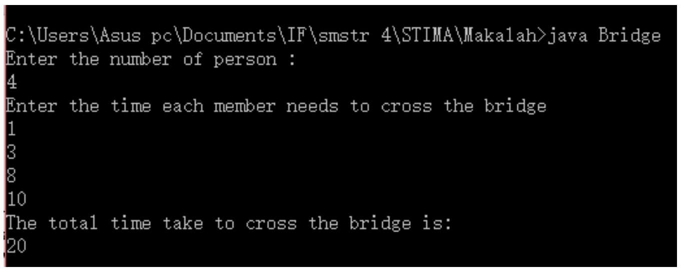
B. Implementation

Speaking of algorithm, it’s such a waste if we know the concept but can’t implement it to the program, so here is the pseudocode of the program :

```
Function TotalTime(person : array of integer, n : integer) ->
integer
variables
        temp1,temp2 : integer
algorithm
        if (n < 3) then
                -> person[n-1]
        else if (n == 3) then
                -> person[0] + person[1] + person[2]
        else
                temp1 = person[n-1] + person[0] + person[n
                - 2] + person[0];
                temp2 = person[1] + person[0] + person[n-
                1] + person[1];

                if (temp1 < temp2)then
                        -> temp1 + TotalTime(person, n-2)
                else if (temp2 < temp1)then
                        -> temp2 + TotalTime(person, n-2)
                else
                        -> temp2 + TotalTime(person, n-2)
end
```

```
Program BridgeandTorch
variables
        n,member : integer
        person : array of integer
algortihm
        write("Enter the number of person : ")
        Read(n)

        write("Enter the time each member needs to cross the
        bridge");

        for(member = 0; member < n ; member++)
                person[member] = input
        endfor

        Sort(person) //sorting time needed to cross the bridge
        write("The total time take to cross the bridge is: ");
        write(Bridge.TotalTime(person, n));

end
```

And this is the following result of the program :



```
C:\Users\Asus pc\Documents\IF\smstr 4\STIMA\Makalah>java Bridge
Enter the number of person :
4
Enter the time each member needs to cross the bridge
1
3
8
10
The total time take to cross the bridge is:
20
```

Figure 3 Implementation of user code

V. Future Implementation

As we can see, the program above still very simple and only determine the optimal time, for the future implementation maybe author will add some code that can show us the process and the solution from step 1 to step k in details.

VI. Conclusion

Dynamic Programming always give an optimal solution for the Bridge and Torch problem. There are 2 approaches of Dynamic programming for solving this problem : Solve by using table and solve by reducing it into Shortest path problem. Both algorithm will give the same and optimal result but the first approach still can be better in performance because it needs smaller amount of memories than the second approach that use graph.

VII. Acknowledgement

The author thanks to Mr. Rinaldi Munir and Mrs. Nur Ulfa Maulidevi for their teaching and endorsement in the IF2211 Algorithms' Strategy course during this semester. The author also thanks to his friends and fellow students of Informatics/Computer Science ITB for their assistance all this time.

References

[1] Munir,Rinaldi. *Slide Kuliah Program Dinamis (2015)*. 2015. Accessed 3 May 2016.

[2] http://www.moshe-online.com/tutor/bridge/. Accessed 5 May 2016

[3] Solving bridge and torch puzzle with Dynamic Programming. http://stackoverflow.com/questions/25399477/solving-bridge-and-torch-puzzle-with-dynamic-programming. Accessed 5 May 2016

[4] https://page.mi.fu-berlin.de/rote/Papers/pdf/Crossing+the+bridge+at+night.pdf. Accessed 6 May 2016

Statement

I hereby declare that the paper I wrote is my own work. It is not a copy nor a translation of someone else's paper, and not a plagiarism.

Bandung, May 7-th 2016

[illegible]

Varian Caesar | 13514041