1 Operations Research (OR):

A scientific approach (mathematical modeling) to decision making. Involves optimization of objective(s) subject to constraint(s). Objective: To make (near-) optimal decisions that maximizes reward or minimizes cost. Reward/cost units: Measured in terms of ($), time, personnel, weight (lbs) or distance (miles) etc.

2 LP vs Deterministic DP

LP is very commonly used, fast, easy to implement, and very good for quick approximate solution (caution!).

Time is an important factor that will affect the model. LP is good for problems that are deterministic, one-time decision making problem at a given time and need the following assumptions. 1. Proportionality: This is guaranteed if the objective and constraints are linear, 2. Additive: Independent decision variables, 3. Divisibility: Fractions allowed, and 4. Certainty: Coefficients in the objective function and constraints must be fixed.

What happens if the following occur?

1. If the problem had to be solved over time, time between decisions is small, there is no computing power to solve large sized problems in that small time interval to optimality.
2. If the problem had stochastic elements such as a probabilistic demand or return on investment.
3. If the proportionality assumption does not hold, that is objective and constraints are nonlinear

The above could be answered with Dynamic Programming.

3 Dynamic Programming

DP is used for sequential decision making. DP is classified as deterministic and stochastic and each of them is further classified as finite and infinite horizon problems.

Finite Horizon- Shortest Path Algorithm

Infinite Horizon- Control Problems

4 DP-Deterministic-Finite Horizon

4.1 Shortest Path Algorithm

Initially will deal with problems in which there are no cycles (called acyclic networks) and the arcs are unidirectional. This is often the case because sequential decisions made over time are unidirectional. Applications: Mapquest, Google Maps, inventory control, equipment replacement, and so on.

Example 2. Find the shortest path in Figure 1.

An arc connects state i with state j and has a value of $c_{i,j}$. A Myopic action is one in which the shortest path is chosen among all the paths that are available from a given state i.

Exhaustive enumeration is not an option due to high computational burden.

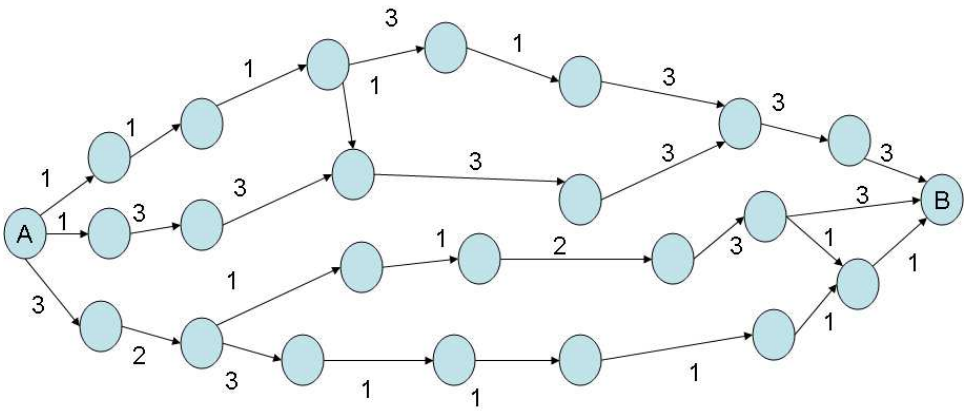

Figure 1: Example 2: Shortest Path.

5 IP vs DP

Some notes on Integer Programming (IP) and Non-linear Programming (NLP).

1. IP is a class of optimization problems in which all the variables are integers. For example, variables that are binary (0 or 1) in an assignment problem of n jobs and m machines. The question is which job goes on which machine and in which order. Also precedence relationship between the operations on the job must be maintained when a job requires more than 1 machine to process.

2. IP is harder to solve since the feasible region has to be searched and not just the extreme points as in LP. This means that the solution need not lie only on the boundary of the feasible region.

3. To search inside the feasible region, you will have to make a start at some point. This will determine how long it will take to reach the solution.

4. There are several search methods which fall under the field of evolutionary computational algorithms (also called meta heuristics) such as branch and bound, tabu search, simulated annealing, and genetic algorithm to solve IP.

5. Mixed integer programming is a special case of IP in which some variables are non-integers.

6. A quick solution to IP is by LP relaxation to IP in which the variables are all relaxed to be non-integers. The solution is than rounded off to the nearest integer. 2 points of CAUTION: 1) rounding off might take the solution outside the feasible area, so care must be taken to ensure that the rounding off is done in such as way that the solution in inside the feasible region by testing the constraints. 2) IP will give incorrect results if LP relaxation is used on binary variables. For example, if your binary decision variable (0-do not invest, 1-invest) turns out to be 0.5 then what will you do? What if it is 0.49 or 0.51 will you round them off to 0 and 1 respectively?

7. Non-linear programming is even harder and has two classes of problems: Constrained and unconstrained optimization. It uses gradient based search techniques which are obtained form the derivatives of the nonlinear function at a point. Functions: convex, concave. Concept of

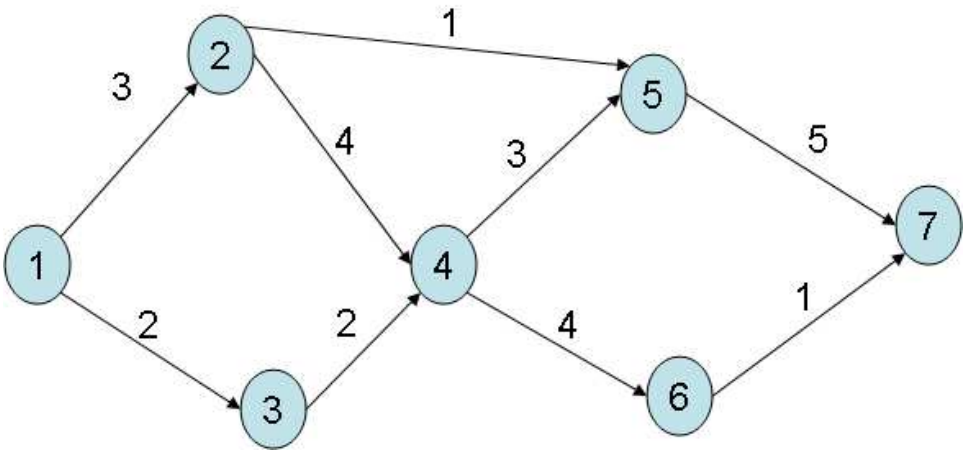

Figure 2: Example 3: Shortest Path.

saddle point, local and global optimum. Solution techniques: Lagrange multiplier, method of steepest descent, Kuhn-Tucker conditions and so on.

5.1 An IP solution to shortest path:

Example 3. Find the shortest path in Figure 2.

Let $x_{ij} = 0$ if arc i, j is not chosen and $x_{ij} = 1$ otherwise. To write the following see Figures 3 and 4

$$Min\ Z = 3x_{12} + 2x_{13} + 1x_{25} + 4x_{24} + 2x_{34} + 3x_{45} + 4x_{46} + 5x_{57} + 1x_{67} \tag{1}$$

subject to

$$\begin{aligned}
x_{12} + x_{13} &= 1 && (2)\\
x_{57} + x_{67} &= 1 && (3)\\
x_{25} + x_{45} + x_{46} &= 1 && (4)\\
x_{25} + x_{24} + x_{34} &= 1 && (5)\\
x_{57} - x_{25} - x_{45} &= 0 && (6)\\
x_{67} - x_{46} &= 0 && (7)\\
x_{25} - x_{12} &\leq 0 && (8)\\
x_{24} - x_{12} &\leq 0 && (9)\\
x_{34} - x_{13} &= 0 && (10)\\
x_{45} - x_{24} - x_{34} &\leq 0 && (11)\\
x_{46} - x_{24} - x_{34} &\leq 0 && (12)\\
x_{24} - x_{12} &\leq 0 && (13)\\
x_{34} - x_{13} &= 0 && (14)\\
x_{ij} &= \textit{binary integers},\ \forall\{ij\} && (15)
\end{aligned}$$

Lindo program code in Figure 5. Lindo solution is in Figure 6

Observations from Example 3

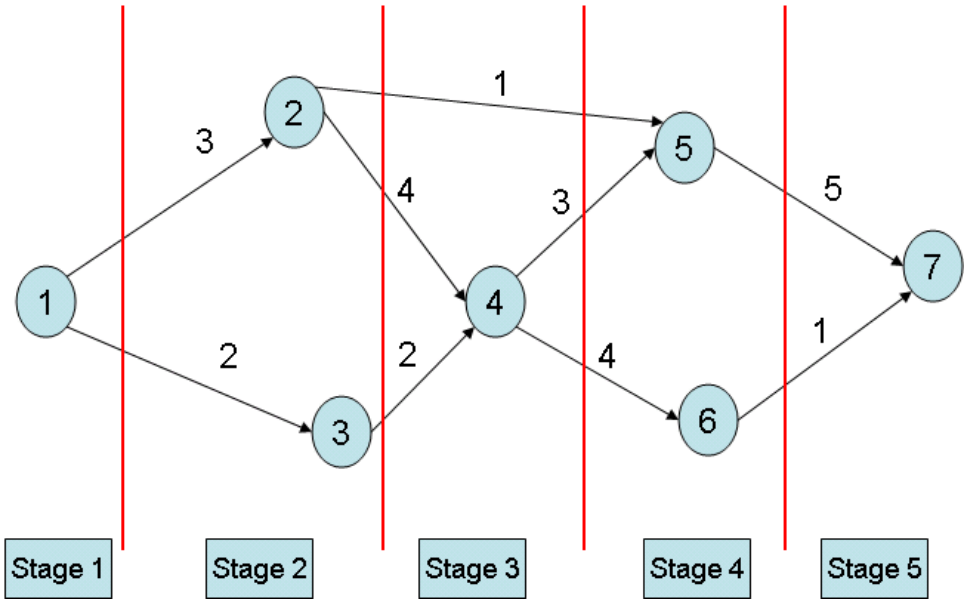

Figure 3: Example 3 with stages marked.

x57	x25	x45
1	1	0
1	0	1
0	0	0

This gives x57-x25-x45=0

x67	x46
1	1
0	0

x67-x46=0

x25	x12
1	1
0	0
0	1

x25-x12<=0

x45	x24	x34
1	0	1
1	1	0
0	0	1
0	1	0

x45-x24-x34<=0

x46	x24	x34
1	0	1
1	1	0
0	0	1
0	1	0

x46-x24-x34<=0

x24	x12
1	1
0	0
0	1

x24-x12<=0

x34	x13
0	0
1	1

x34-x13=0

Figure 4: Example 3: Steps to form constraints.

```
min 3 x12 + 2 x13 + 1 x25 + 4 x24 + 2 x34 + 3 x45 + 4 x46+ 5
x57 + 1 x67
st
x57-x25-x45=0
x67-x46=0
x25-x12<=0
x24-x12<=0
x34-x13=0
x12+x13=1
x57+x67=1
x25+x45+x46=1
x25+x24+x34=1
x45-x24-x34<=0
x46-x24-x34<=0
end
int x12
int x13
int x25
int x24
int x34
int x45
int x46
int x57
int x67
```

Figure 5: Example 3 Lindo code.

Writing IP equations as given in Figure 4 is not easy as the number of state increases. Also IP finds only one optimal solution even though example 3 has 2 solutions. DP solution is the fastest way to solve this problem and it finds both solutions.

5.2 Myopic Solution

A myopic policy does not yield optimal solution. This is because its actions are dependent on the contribution function and not the value function.

```
LP OPTIMUM FOUND AT STEP      5
 OBJECTIVE VALUE =   9.00000000

 NEW INTEGER SOLUTION OF    9.00000000     AT BRANCH      0
PIVOT       5
 RE-INSTALLING BEST SOLUTION...

       OBJECTIVE FUNCTION VALUE

       1)      9.000000

  VARIABLE        VALUE          REDUCED COST
       X12         0.000000          3.000000
       X13         1.000000          2.000000
       X25         0.000000          1.000000
       X24         0.000000          4.000000
       X34         1.000000          2.000000
       X45         0.000000          3.000000
       X46         1.000000          4.000000
       X57         0.000000          5.000000
       X67         1.000000          1.000000


      ROW   SLACK OR SURPLUS     DUAL PRICES
       2)          0.000000          0.000000
       3)          0.000000          0.000000
       4)          0.000000          0.000000
       5)          0.000000          0.000000
       6)          0.000000          0.000000
       7)          0.000000          0.000000
       8)          0.000000          0.000000
       9)          0.000000          0.000000
      10)          0.000000          0.000000
      11)          1.000000          0.000000
      12)          0.000000          0.000000

 NO. ITERATIONS=       5
 BRANCHES=    0 DETERM.=  1.000E    0
```

Figure 6: Example 3: Lindo solution

5.3 DP solution to Example 3:

Terminology used in DP: State i, j, Stage t, action or decision, k, optimal policy R, value function of a state which is denoted as f_i or V_i or $f(i)$ or $V(i)$.

Fundamental Mathematical Construction in DP is the RECURSIVE EQUATION

A recursive equation is a relation between t and $t+1$ along with an optimization operator (max or min or minmax or maxmin). DP is solved backward for finite horizon problems. Recursive relation formalizes this backward procedure.

5.3.1 Steps in Backward Recursion

1. Step 1. Let f_i be the value function at state i.
2. Step 2. Let $f_{last-state}$ =0;
3. Step 3.

$$f_i = \min_x [c_{ix} + f_j],\ i < j \tag{16}$$

Or,

4. Step 4. If stage t is specified then

$$f_t(i) = \min_x [c_{ix} + f_{t+1}(j)], \quad \forall i, \tag{17}$$

5. Stop when starting state and stage is reached

6. Get the shortest path by going forward.

Recursive Solution to Example 3

$$\begin{aligned}
f_7 &= 0 && (18)\\
f_6 &= min[c_{67} + f_7] = 1 && (19)\\
f_5 &= min[c_{57} + f_7] = 5 && (20)\\
f_4 &= min[c_{45} + f_5, c_{46} + f_6] = min[8, 5] = 5 && (21)\\
f_3 &= min[c_{34} + f_4] = 7 && (22)\\
f_2 &= min[c_{25} + f_5, c_{24} + f_4] = min[1 + 5, 4 + 5] = 6 && (23)\\
f_1 &= min[c_{12} + f_2, c_{13} + f_3] = min[9, 9] = 9 && (24)
\end{aligned}$$

Tracing forward, the paths are, at 1 both 2 and 3 (1-2, 1-3), at 2 it is 5 (1-2-5), at 3 it is 4 (1-3-4), at 5 it is 7 (1-2-5-7), at 4 it is 6 (1-3-4-6), at 6 it is 7 (1-3-4-6-7).
Solution: (1-2-5-7) and (1-3-4-6-7), path length = 9

6 Observations from the DP solution to Example 3

1. A deterministic finite horizon problem can be solved backwards (tracing the solution forward) or forward (tracing the solution backwards).

2. For all problems (deterministic or stochastic, finite or infinite horizon) we will follow the backward recursion formula because forward reaching will not be applicable to the stochastic infinite case due to the constraints placed by the Markov chain. For finite horizon problems, forward reaching is used only if end state (sink) is not known.

3. Begin solving by setting the value of the last state in the last stage to zero and work backwards till the first state in the first stage.

4. If there is more than one solution to the max or min operator at any state then there are multiple optimal paths.

5. At any single iteration, the calculations of the value function is only between the current state at t and the future states at $t+1$. This has a computational advantage because the algorithm performs only a few calculations at t even though the problem could have millions of states occurring at other times.

6. The value functions at a state are cumulative from the current state at t till the end of the problem in a backward recursion setting. In other words, it is the sum of all the c_{ix}'s starting from state i at t till the end of the problem for a given network that evolves over several stages.

7. The value of the first state in first stage is the solution of the problem or the objective function value.

8. Actions are defined for every stage and for all states in that stage. The collection of optimal actions from the first to the last state/stage gives the optimal path (aka policy R).

9. Feasible actions at a state in a stage will depend on the current state. However, the action MUST influence the state variables. Otherwise keep those variables out of the state and treat them as just observable variables that do not belong to the state.

10. For real world problems, both action and state, and uncertainty in the case of stochastic DP, are all multi-dimensional, which causes computational storage issues known as the curse of dimensionality. Methods are there to mitigate this curse but more research is needed.

11. 2 steps - Model the problem using the DP recursion by defining stage, state, action, contribution function, and exogenous information, then solve the model to obtain the objective function and the optimal action path through the stages t.

12. The objective is to go from one good state at t to another good state at $t+1$ by taking an optimal action at t under uncertainty.

To solve any sequential decision making problem with DP one must identify and define the following elements. The main elements of the DP recursive equation are

1. Stage t: For finite horizon problems it is usually time t (but not always true). For infinite horizon problems its always time.

2. State i or j and in general S: Usually its the AVAILABLE resource that need to be allocated at stage t. However, there are problems where the state is the inventory at hand, or the price of an asset in an asset acquisition problem (e.g. stock market, oil prices), and so on. The next state of a system depends on the current state and the action taken in the current state.

3. Action or decision x or a or k: The action (x_t or a_t) taken in (state i at stage t or S_t) that moves the system to state (j at stage $t+1$ or S_{t+1}) under the influence of a exogenous process W. This means S_{t+1} is a function of (S_t, x_t, W_{t+1}).

4. S_{t+1} is a function of (S_t, x_t, W_{t+1}). Define S_{t+1}, which in turn defines the state transition function.

5. Exogenous process W: Its either deterministic or stochastic (uncertainty). For example, in inventory control problems it is the demand.

6. Contribution function (reward or cost): C or c or r and depending on the type of problem you will use $C(i,a,j)$ or $C(S,x)$ or $c_{i,j}$ and so on. This is the immediate contribution of an action taken in a particular state. Also known as the one-step cost or reward function.

7. Value function of a state f or V: The long-run value $f_t(S)$ of being in a state S at stage t. This is used in making decisions in state S.

8. Transition probability p(i,a,j) or P(i,a,j): This is only for stochastic DP where it denotes the probability of transitioning from state i to state j under action a.

9. Objective function: This is the max or min operator that acts on the value function.

Exercise 2. [1].
Find the shortest path for the problem in Figure 7.

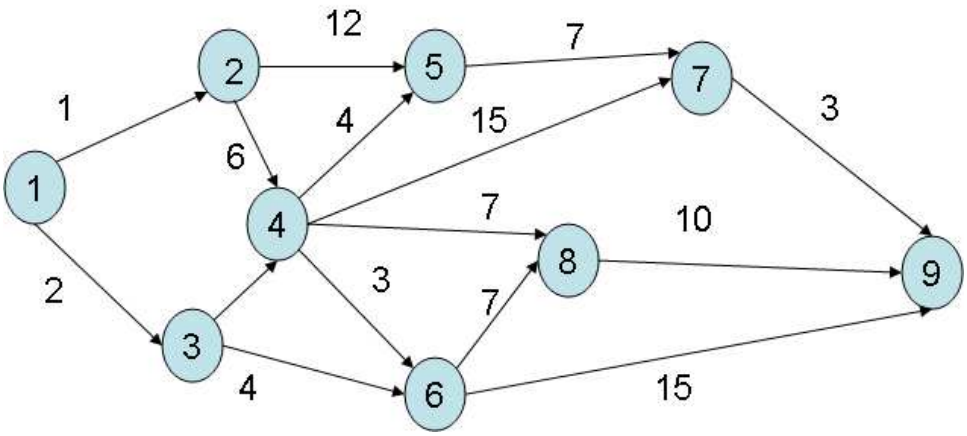

Figure 7: Exercise 2: Shortest Path.

6.1 Shortest Path Problem: Forward Calculation (Reaching)

$$f_j = \min_x [c_{ix} + f_i],\ i < j \tag{25}$$

Or, if stage t is specified then

$$f_{t+1}(j) = \min_x [c_{ix} + f_t(i)],\quad \forall i, \tag{26}$$

Trace the path backwards.
Exercise 3. Solve Example 3 with Reaching to find the shortest path.

6.2 Longest Path Problem: Forward Calculation (Reaching)

Longest Path Algorithms give the critical path of a network. They are useful in applications such as finding the project due date or earliest finish date of the project, or maximizing reward. Replacing min with max
Exercise 4. Solve Example 3 with Reaching to find the longest path.
Exercise 5. Solve Example 3 with backward recursion to find the longest path.

6.3 Computational Efficiency of DP vs exhaustive enumeration

To solve the shortest path in Figure 8, if one has to solve the 5^5 paths explicitly then with 5 additions in each path there are $(5^5) \times 5 = 15625$ additions. With DP, there are $4(25) + 5 = 105$ additions.

6.4 General Recursion Definition for Min problems

Value of being in state i at stage t = min (cost of an action in state i at stage t which takes you to state j at stage $t+1$ plus the value of being in state j at stage $t+1$.

6.5 Dijkstra's Algorithm for shortest path

See Figure 9. V= All vertices (states), and S is an empty set. Acquire nodes into S until

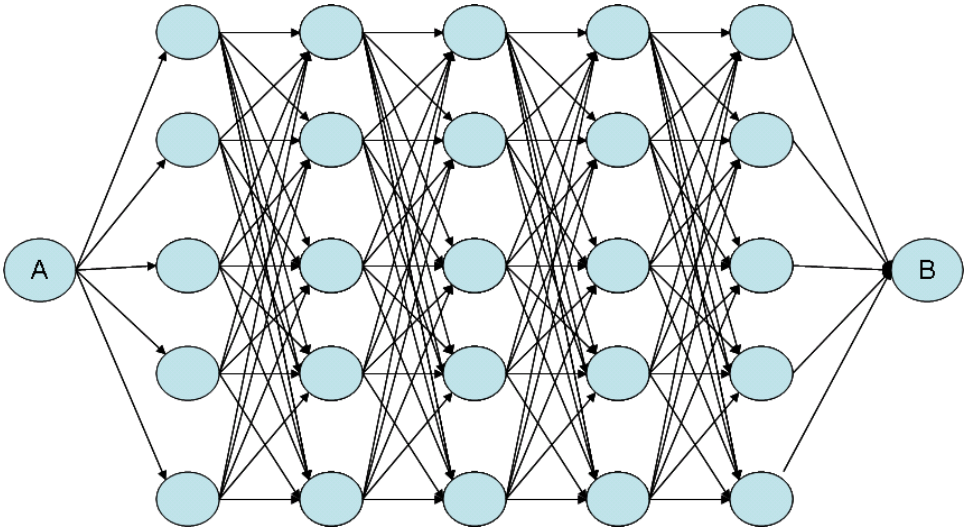

Figure 8: Computational Efficiency.

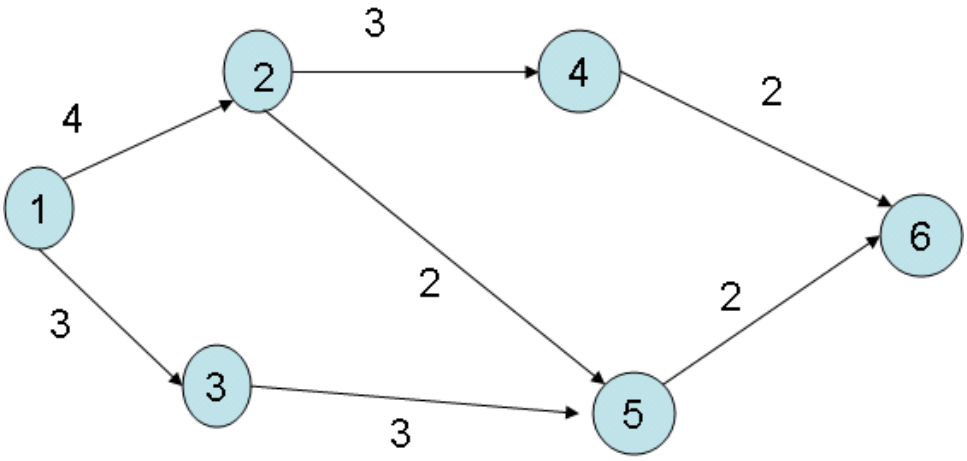

Figure 9: Dijkstra's Algorithm.

$$V - S = \phi \ (Null\ Set) \tag{27}$$
$$f_1 = 0 \tag{28}$$

Update all f_i at every step. Initialize all f's except f_1 to ∞

1. Step 1. Closest neighbor to 1 is 3, so acquire 3

$$S = (1,3) \tag{29}$$
$$f_3 = 3 + f_1 = 3 \tag{30}$$
$$f_2 = 4 + f_1 = 4 \tag{31}$$

3 has 1 as predecessor

2. Step 2. Next closest neighbors to S are

$$\begin{aligned} f_2 &= 4 + f_1 = 4 \quad Min\ so\ acquire\ 2 && (32)\\ f_5 &= 3 + f_3 = 6 && (33)\\ S &= (1, 3, 2) && (34) \end{aligned}$$

   2 has 1 as predecessor

3. Step 3. Next closest neighbors to S are:

$$\begin{aligned} f_4 &= f_2 + 3 = 7 && (35)\\ f_5 &= \min(2 + f_2, 3 + f_3) = \min(6, 6) = 6 \quad Min\ so\ acquire\ 5 && (36)\\ S &= (1, 2, 3, 5) && (37) \end{aligned}$$

   5 has 2 and 3 as predecessor

4. Step 4. Next closest to S are

$$\begin{aligned} f_4 &= f_2 + 3 = 7 \quad Min\ so\ acquire\ 4 && (38)\\ f_6 &= \min(2 + f_5) = \min(8) = 8 && (39)\\ S &= (1, 2, 3, 5, 4) && (40) \end{aligned}$$

   4 has 2 as predecessor

5. Step 5. Acquire 6

$$f_6 = \min(2 + f_4, 2 + f_5) = \min(9, 8) = 8 \quad (41)$$

   6 has 5 as predecessor

Tracing backwards using predecessor relationship
2 possible solutions (1-2-5-6), (1-3-5-6)
Total time = 8 =f_6

6.6 Matlab code for the length of the shortest path

See Figure 10

```
c=[0 3 2 0 0 0 0
   0 0 0 4 1 0 0
   0 0 0 2 0 0 0
   0 0 0 0 3 4 0
   0 0 0 0 0 0 5
   0 0 0 0 0 0 1
   0 0 0 0 0 0 0]
count=0;
temp=0;
d=0;
f=0;
D=size(c);
f(D(1,1),1)=0;
for i= D(1,1)-1:-1:1
    count=0;
    for j=i:1: D(1,1)
         if c(i,j) ~= 0
             count=count+1;
             temp(count,1)= c(i,j)+f(j,1);
         end
    end
    f(i,1)=min (temp (:,1));
    temp=0;
end
f
```

Figure 10: Matlab code for length of Shortest Path.

References

[1] E.V. Denardo. *Dynamic Programming: Models and Applications*. Dover Publications, 2003.