Chapter 4

Shortest Paths

4.1 Introduction

Shortest Path Problem (SPP) allure researchers and practitioners for several reasons: (i) they arise in a lot of applications, (ii) they can be solved efficiently, (iii) they capture the core ingredients of many NFP, and (iv) they frequently arise as sub-problems when solving other NFP.

We consider a directed graph $G = (N, A)$ with an *arc cost* (or *arc length*) c_{ij} associated with each arc $(i, j) \in A$. The graph has n nodes (i.e., $|N| = n$) and a distinguished node $s \in N$, called the *source*. Let $C = \max\{c_{ij} : (i, j) \in A\}$; moreover, let $\delta_i^+ \subseteq N$ ($\delta_i^- \subseteq N$, respectively) the set of successors (predecessors, resp.) of node $i \in N$, i.e., $\delta_i^+ = \{j \in N \,|\, (i, j) \in A\}$ ($\delta_i^- = \{j \in N \,|\, (j, i) \in A\}$, resp.). We define the *length of a directed path* as the sum of the lengths of arcs in the path. The SPP is to determine, for every node $i \in N \setminus \{s\}$, a shortest length directed path from node s to node i.

By introducing continuous variables $x_{ij} \in \mathbb{R}_+$ representing the flow through arc $(i, j) \in A$, the SPP can be formulated as follows

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{4.1a}$$

$$\text{s.t.} \sum_{j\in\delta_s^+} x_{sj} = n - 1 \tag{4.1b}$$

$$\sum_{i\in\delta_k^-} x_{ik} - \sum_{j\in\delta_k^+} x_{kj} = 1 \quad \forall k \in N \setminus \{s\} \tag{4.1c}$$

$$x_{ij} \in \mathbb{R}_+ \qquad \forall (i, j) \in A \tag{4.1d}$$

The objective function (4.1a) aims at minimizing the total cost of the selected paths. Constraint (4.1b) ensures that $n - 1$ units of commodity emanate from the source node. Constraints (4.1c) guarantee that a unit of commodity is intended for each non-source node. Constraints (4.1d) define the range of the decision variables.

Assumption 4.1. *All arc lengths are integers.*

Some algorithms need the integrality assumption. Others do not. Note that rational arc lengths can always be transformed to integer arc lengths by multiplying them by a suitably large number. Irrational numbers need to be converted to rational numbers to represent them on a computer. The integrality assumption is not restrictive in practice.

Assumption 4.2. *The graph contains a directed path from node s to every other node in the graph.*

This assumption can always be satisfied by adding a "fictitious" arc (s, i) of suitably large cost for each node i not connected to s by a directed path.

Assumption 4.3. *The graph does not contain a negative cycle (i.e., a directed cycle of negative length).*

Without this assumption, formulation (4.1) has an unbounded solution because an infinite flow can be sent through a negative cycle, and the SPP becomes significantly harder to solve.

Assumption 4.4. *The graph is directed.*

If the graph is undirected, it can be suitably transformed in a directed graph.

4.2 Applications

SPP arise in a wide variety of practical problems settings, both as stand-alone models and as subproblems in more complex problem settings. For example, they arise in telecommunications, transportation, urban traffic planning, project management, inventory planning, DNA sequencing, etc.

Exercise 4.1.

A production line consists of an ordered sequence of n production stages. Each stage has a manufacturing operation followed by a potential inspection. The products enter the production line in batches of size $B \geq 1$. As the items within a batch move through the manufacturing stages, the operations might introduce defects. The probability of producing a defect at stage i is α_i. Defects are not repairable, so we must scrap any defective item. After each stage, we can either inspect all of the items or none of them (we do not sample the items). The inspection identifies every defective item. The production line must end with an inspection station so that we do not ship any defective units.

Our decision-making problem is to find an optimal inspection plan that specifies at which stages we should inspect the items so that we minimize the total cost of production and inspection. The following cost data are available: (i) p_i, the manufacturing cost per unit in stage i; (ii) f_{ij}, the fixed cost of inspecting a batch after stage j given that we last inspected the batch after stage i; and (iii) g_{ij}, the variable unit cost for inspecting an item after stage j, given that we last inspected the batch after stage i.

How can this decision-making problem be formulated as a SPP?

We can formulate this inspection problem as a SPP on a graph with $n + 1$ nodes ($N = \{0, 1, \dots, n\}$). The graph contains an arc (i, j) for each pair of nodes i and j such that $i < j$, i.e., $A = \{(i, j) \mid i, j \in N : i < j\}$. Figure 4.1 shows the graph for an inspection problem with four stations. Each path in the graph from node 0 to node 4 defines an inspection plan. For example, the path $0 - 2 - 4$ implies that we inspect the batches after the second and the fourth stages.

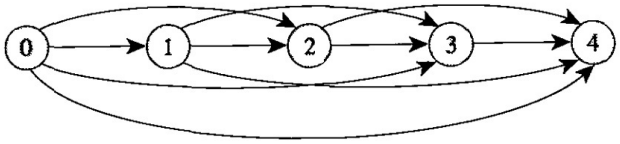


Figure 4.1: Graph of an inspection problem with four stations

The inspection problem can be solved as a SPP on such a graph, where costs c_{ij} are defined for each arc $(i, j) \in A$ as follows. Let $b_i = B\prod_{k=1}^{i}(1 - \alpha_k)$ denote the expected number of non-defective units at the end of stage i. Costs c_{ij} are defined as

$$c_{ij} = f_{ij} + b_i g_{ij} + b_i \sum_{k=i+1}^{j} p_k \qquad \forall (i, j) \in A$$

Exercise 4.2.

A hiker must decide which goods to include in her knapsack on a trip. She must choose from among p objects. Object i has a weight w_i (in kilos) and a utility u_i to the hiker. The objective is to maximize the utility of the hiker's trip subject to the weight limitation that she can carry no more than W kilos. How can this decision-making problem be formulated as a SPP?

We can define a graph with $p(W + 1) + 2$ nodes. The node set N contains two dummy nodes s and t plus $p(W + 1)$ nodes i^k (with $i = 1, \dots, p$ and $k = 0, 1, \dots, W$) representing the decision to use k kilos of capacity to carry a subset of the first i objects. The arc set A is defined as

$$A = A^s \cup A^0 \cup A^1 \cup A^t,$$

where

$$\begin{aligned}
A^s &= \{(s, 1^0), (s, 1^{w_1})\} \\
A^0 &= \{(i^k, (i+1)^k) \mid i = 1, \dots, p-1,\ k = 0, 1, \dots, W\} \\
A^1 &= \{(i^k, (i+1)^{k+w_{i+1}}) \mid i = 1, \dots, p-1,\ k = 0, 1, \dots, W : k + w_{i+1} \le W\} \\
A^t &= \{(p^k, t) \mid k = 0, 1, \dots, W\}
\end{aligned}$$

The cost c_{ij} of each arc $(i, j) \in A$ is defined as

$$c_{ij} = \begin{cases} 0 & \text{if } (i,j) \in \{(s, 1^0)\} \cup A^0 \cup A^t \\ -u_1 & \text{if } (i,j) = (s, 1^{w_1}) \\ -u_j & \text{if } (i,j) \in A^1 \end{cases}$$

Finding a shortest path from s to t corresponds to finding the highest-utility subset of objects to select, where traversing an arc (i, j) with a strictly negative cost means that object j is selected and traversing an arc (i, j) with a zero cost means that object j is not selected.

Consider the following example: $p = 4$, $W = 6$, $\boldsymbol{u} = \{40, 15, 20, 10\}$, $\boldsymbol{w} = \{4, 2, 3, 1\}$. The corresponding graph is depicted in Figure 4.2, where the number close to each arc is the opposite of

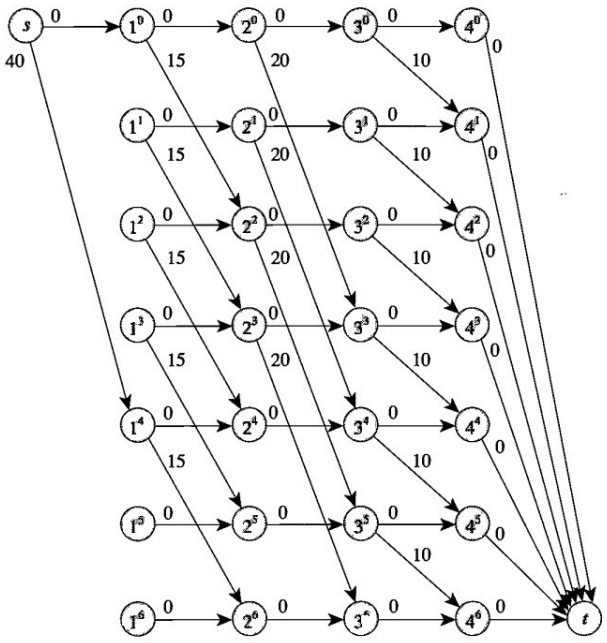


Figure 4.2: Graph representation of the example of the knapsack problem

its costs (i.e., $-c_{ij}$). The shortest path from s to t is $s - 1^4 - 2^6 - 3^6 - 4^6 - t$, corresponding to selecting objects 1 and 2 with a cost of −55 (i.e., a utility of 55 for the hiker).

This problem is known as the $0-1$ knapsack problem. Let $y_i \in \{0, 1\}$ be a binary variable equal to 1 if object i ($i = 1, \dots, p$) is selected (0 otherwise). The problem can be formulated as

$$\max \sum_{i=1}^{p} u_i y_i \tag{4.2a}$$

$$\text{s.t.} \sum_{i=1}^{p} w_i y_i \leq W \tag{4.2b}$$

$$y_i \in \{0, 1\} \qquad \forall i = 1, \dots, p \tag{4.2c}$$

4.3 Tree of Shortest Paths

In the SPP, we wish to determine a shortest path from the source node to all other $(n-1)$ nodes. How much storage do we need to store these paths? A naive answer is $(n-1)^2$ storage locations since each path contain at most $(n-1)$ arcs. Fortunately, $(n-1)$ storage locations are sufficient. Indeed, we can always find a tree rooted from the source with the property that the unique path from s to any node is a shortest path to that node.

Property 4.1. *If the path $s = i_1 - i_2 - \cdots - i_h = k$ is a shortest path from node s to node k, then for every $q = 2, 3, \dots, h-1$, the subpath $s = i_1 - i_2 - \cdots - i_q$ is a shortest path from s to node i_q.*

Proof. We provide an informal proof. In Figure 4.3, we assume that the shortest path $P_1 - P_3$ from s to k passes through a node p, but the sub-path P_1 up to node p is not a shortest path to node p. Suppose

instead that path P_2 is shorter than P_1. Then path $P_2 - P_3$ is shorter than $P_1 - P_3$, which contradicts the assumption of optimality of path $P_1 - P_3$. Therefore, P_1 must be a shortest path from s to p.∎

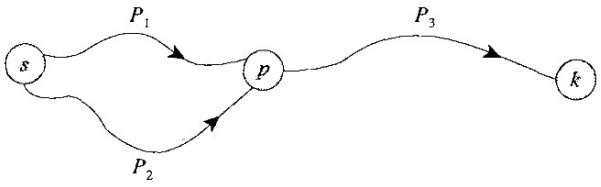


Figure 4.3: Optimality of sub-paths of shortest paths

Let $d(\cdot)$ denote the shortest path distances. The previous property implies that, if P is a shortest path from s to k, then $d(j) = d(i) + c_{ij}$ for every arc $(i,j) \in P$. The converse is also true: if $d(j) = d(i) + c_{ij}$ for every arc in a directed path P from s to node k, then P must be a shortest path.

Indeed, we can observe that

$$\begin{aligned} d(k) &= d(i_h) \\ &= d(i_{h-1}) + c_{i_{h-1}i_h} = \\ &= d(i_{h-2}) + c_{i_{h-2}i_{h-1}} + c_{i_{h-1}i_h} = \\ &= d(i_{h-3}) + c_{i_{h-3}i_{h-2}} + c_{i_{h-2}i_{h-1}} + c_{i_{h-1}i_h} = \\ &\vdots \\ &= d(i_1) + c_{i_1 i_2} + c_{i_2 i_3} + \ldots + c_{i_{h-1}i_h} = \\ &= 0 + \sum_{(i,j)\in P} c_{ij} \end{aligned}$$

Consequently, P is a directed path from s to k of length $d(k)$. Since, by assumption, $d(k)$ is the shortest path distance to k, P is a shortest path to k.

Property 4.2. *Let the vector* $\mathbf{d}$ *represent the shortest path distances. A directed path* P *from* s *to* k *is a shortest path if and only if* $d(j) = d(i) + c_{ij}$ *for every arc* $(i,j) \in P$.

4.4 Dijkstra's Algorithm

The Dijkstra's algorithm finds shortest paths from source node s to all other nodes in a graph with non-negative arc lengths. It maintains a distance label $d(i)$ with each node i, which is a UB on the shortest path length to node i, and the predecessor of node i ($pred(i)$) in the shortest path from s. At any intermediate step, the algorithm divides the nodes into two groups: *permanent* nodes (P) and *temporary* nodes (T). The distance label to any permanent node is the shortest distance from s to that node. For any temporary node, the distance label is a UB on the shortest path distance to that node.

The basic idea of the algorithm is to fan out from s and permanently label nodes in the order of their distances from s. Initially, s is given a permanent label of zero, and each other node j a temporary label of ∞. At each iteration, the algorithm selects a node i with the minimum temporary label (breaking ties arbitrarily), makes it permanent, and reaches out from the node - that is, scans arcs emanating from i to update the distances of the adjacent nodes. The algorithm terminates when

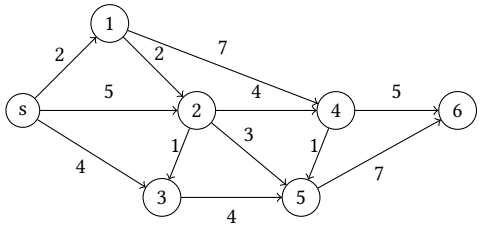
Algorithm 1: Dijkstra's Algorithm

```
1  P ← ∅; T ← N;
2  d(i) ← ∞ for each i ∈ N;
3  d(s) ← 0 and pred(s) ← 0;
4  while P ≠ N do
5      Let i ∈ T be a node for which d(i) = min{d(j) : j ∈ T};
6      P ← P ∪ {i}; T ← T \ {i};
7      for each (i, j) ∈ A do
8          if d(j) > d(i) + c_ij then
9              d(j) ← d(i) + c_ij;
10             pred(j) ← i;
```

all nodes are permanent. The pseudocode of the Dijkstra's algorithm is provided in Algorithm 1 – we do not prove the correctness of the algorithm, which can be established with inductive arguments.

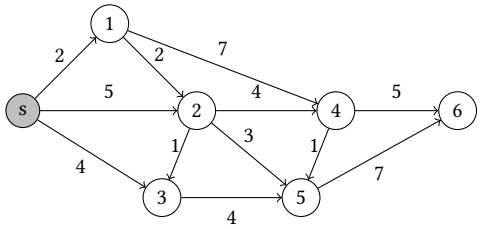
We illustrate the Dijkstra's algorithm with the following example.

- Initialization



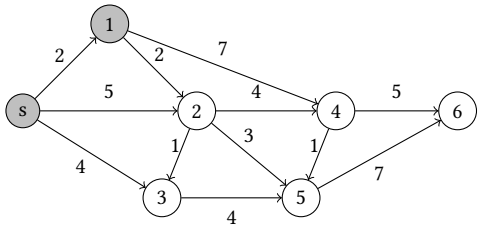
j	s	1	2	3	4	5	6
$d(j)$	0	∞	∞	∞	∞	∞	∞
$pred(j)$	0	-	-	-	-	-	-
i =? $P = \emptyset$ $T = \{s, 1, 2, 3, 4, 5, 6\}$							

- Iteration 1



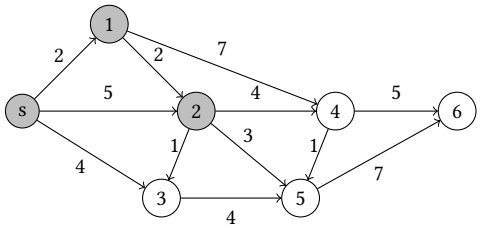
j	s	1	2	3	4	5	6
$d(j)$	0	2	5	4	∞	∞	∞
$pred(j)$	0	s	s	s	-	-	-
$i = s$ $P = \{s\}$ $T = \{1, 2, 3, 4, 5, 6\}$							

- Iteration 2



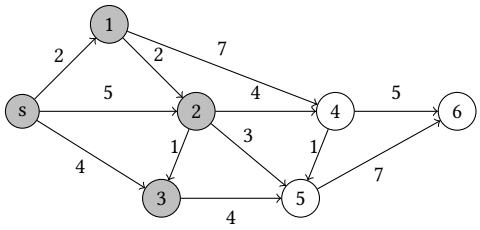
j	s	1	2	3	4	5	6
$d(j)$	0	2	4	4	9	∞	∞
$pred(j)$	0	s	1	s	1	-	-
$i = 1$ $P = \{s, 1\}$ $T = \{2, 3, 4, 5, 6\}$							

- Iteration 3



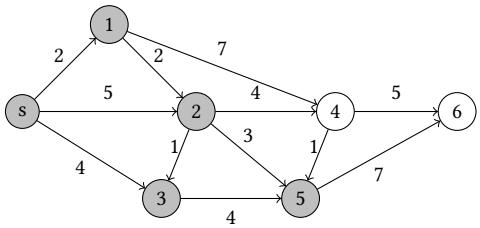
j	s	1	2	3	4	5	6
$d(j)$	0	2	4	4	8	7	∞
$pred(j)$	0	s	1	s	2	2	-
$i = 2$ $P = \{s, 1, 2\}$ $T = \{3, 4, 5, 6\}$							

- Iteration 4



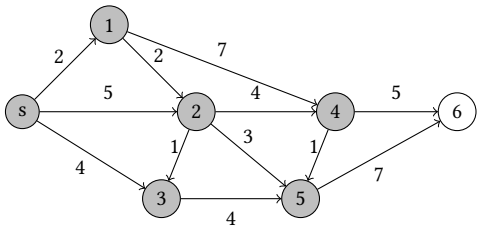
j	s	1	2	3	4	5	6
$d(j)$	0	2	4	4	8	7	∞
$pred(j)$	0	s	1	s	2	2	-
$i = 3$ $P = \{s, 1, 2, 3\}$ $T = \{4, 5, 6\}$							

- Iteration 5



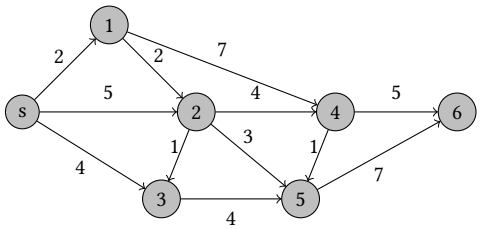
j	s	1	2	3	4	5	6
$d(j)$	0	2	4	4	8	7	14
$pred(j)$	0	s	1	s	2	2	5
$i = 5$ $P = \{s, 1, 2, 3, 5\}$ $T = \{4, 6\}$							

- Iteration 6



j	s	1	2	3	4	5	6
$d(j)$	0	2	4	4	8	7	13
$pred(j)$	0	s	1	s	2	2	4
$i = 4$ $P = \{s, 1, 2, 3, 4, 5\}$ $T = \{6\}$							

- Iteration 7



j	s	1	2	3	4	5	6
$d(j)$	0	2	4	4	8	7	13
$pred(j)$	0	s	1	s	2	2	4
$i = 6$ $P = \{s, 1, 2, 3, 4, 5, 6\}$ $T = \emptyset$							

Exercise 4.3.

Consider a set of n scalar numbers $a_1, a_2, \dots, a_n$ arranged in non-decreasing order of their values. We wish to partition these numbers into clusters so that (i) each cluster contains at least p numbers, (ii) each cluster contains consecutive numbers from the list, and (iii) the sum of the squared deviation of the numbers from their cluster means is as small as possible. Let $m(S) = \sum_{i \in S} a_i/|S|$ denote the mean of a set S of numbers defining a cluster. If the number a_k belongs to cluster S, the squared deviation of a_k from the cluster mean is $(a_k - m(S))^2$.

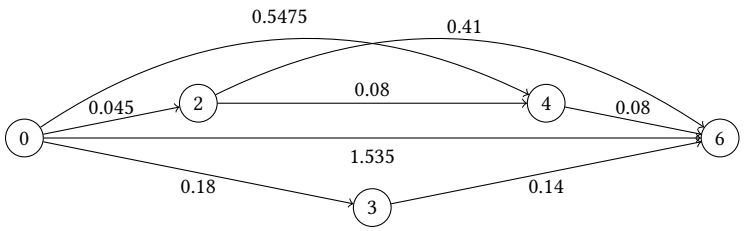
Formulate this cluster analysis problem, and provide an optimal solution of the following case: $p = 2$, $n = 6$, $a_1 = 0.5$, $a_2 = 0.8$, $a_3 = 1.1$, $a_4 = 1.5$, $a_5 = 1.6$, and $a_6 = 2.0$.

This cluster analysis problem can be formulated and solved as a SPP on the following graph $G = (N, A)$. The set of nodes N contains $n + 1$ nodes and is defined as $N = \{0, 1, \dots, n\}$. The node 0 is a dummy node, and the other n nodes correspond to the n scalars. The arc set is defined as $A = \{(i, j) \mid i, j \in N : i + p \le j\}$. Each arc $(i, j) \in A$ represents a cluster of scalars containing all scalars from $i+1$ to j. The cost c_{ij} of arc $(i, j) \in A$ is computed as $\sum_{k=i+1}^{j} (a_k - m_{ij})^2$, where $m_{ij} = \sum_{k=i+1}^{j} a_k/(j - i)$. An optimal clustering of the scalars corresponds to a shortest path from node 0 to node n.

In the case, we have $N = \{0, 1, \dots, 6\}$ and $A = \{(0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 3), (1, 4), (1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 5), (3, 6), (4, 6)\}$. Notice that some arcs can be ruled out as it does not exist any paths from 0 to 6 traversing them, namely, $(0, 5), (1, 3), (1, 4), (1, 5), (1, 6), (2, 5), (3, 5)$. This observation allows to reduce the size of A from 15 to 8, so we focus on a reduced arc set $A = \{(0, 2), (0, 3), (0, 4), (0, 6), (2, 4), (2, 6), (3, 6), (4, 6)\}$. Let us compute the cost of each arc

- $(0, 2)$: $m_{02} = (0.5 + 0.8)/2 = 0.65$, $c_{02} = (0.5 - 0.65)^2 + (0.8 - 0.65)^2 = 0.045$
- $(0, 3)$: $m_{03} = (0.5 + 0.8 + 1.1)/3 = 0.8$, $c_{03} = (0.5 - 0.8)^2 + (0.8 - 0.8)^2 + (1.1 - 0.8)^2 = 0.18$
- $(0, 4)$: $m_{04} = (0.5 + 0.8 + 1.1 + 1.5)/4 = 0.975$, $c_{04} = (0.5 - 0.975)^2 + (0.8 - 0.975)^2 + (1.1 - 0.975)^2 + (1.5 - 0.975)^2 = 0.5475$
- $(0, 6)$: $m_{06} = (0.5 + 0.8 + 1.1 + 1.5 + 1.6 + 2)/6 = 1.25$, $c_{06} = 1.535$
- $(2, 4)$: $m_{24} = (1.1 + 1.5)/2 = 1.3$, $c_{24} = 0.08$
- $(2, 6)$: $m_{26} = (1.1 + 1.5 + 1.6 + 2.0)/4 = 1.55$, $c_{26} = 0.41$
- $(3, 6)$: $m_{36} = (1.5 + 1.6 + 2.0)/3 = 1.7$, $c_{36} = 0.14$
- $(4, 6)$: $m_{46} = (1.6 + 2.0)/2 = 1.8$, $c_{46} = 0.08$

The graph $G = (N, A)$ is



The Dijkstra's algorithm goes through the following iterations

				$d(\cdot)$					$pred(\cdot)$				
Iter	i	P	T	0	2	3	4	6	0	2	3	4	6
0		$\emptyset$	$\{0,2,3,4,6\}$	0	∞	∞	∞	∞	0				
1	0	$\{0\}$	$\{2,3,4,6\}$	0	0.045	0.18	0.5475	1.535	0	0	0	0	0
2	2	$\{0,2\}$	$\{3,4,6\}$	0	0.045	0.18	0.125	0.455	0	0	0	2	2
3	4	$\{0,2,4\}$	$\{3,6\}$	0	0.045	0.18	0.125	0.205	0	0	0	2	4
4	3	$\{0,2,3,4\}$	$\{6\}$	0	0.045	0.18	0.125	0.205	0	0	0	2	4
5	6	$\{0,2,3,4,6\}$	$\emptyset$	0	0.045	0.18	0.125	0.205	0	0	0	2	4

The shortest path is $0-2-4-6$, which corresponds to clustering the scalars in three sets: $\{1,2\}$, $\{3,4\}$, and $\{5,6\}$. The sum of the squared deviations is 0.205.

Exercise 4.4.

A construction company's work schedule on a certain site requires the following number of skilled personnel, called steel erectors, in the months of March through August

Month	Mar	Apr	May	Jun	Jul	Aug
Personnel	4	6	7	4	6	4

Personnel work at the site on the monthly basis. Suppose that three steel erectors are on the site in February and three steel erectors must be on site in September. The problem is to determine how many workers to have on site in each month in order to minimize costs, subject to the following conditions:

- *Transfer Costs*: adding a worker to this site costs €100 per worker and redeploying a worker to another site costs €160
- *Transfer Rules*: the company can transfer no more than three workers at the start of any month, and under a union agreement, it can redeploy no more than one of the current workers in any trade from a site at the end of any month
- *Surplus and Shortage*: the company incurs a cost of €200 per worker per month for having a surplus of steel erectors on site and a cost of €200 per worker per month for having a shortage of workers at the site.

Formulate this personnel planning problem as a SPP.

Let us introduce the following notation to refer to the input data:

K set of months (i.e., $K = \{2,3,4,5,6,7,8,9\}$, months from 2 – February until 9 – September)

w_k workers required at the end of month $k \in K$

W maximum number of workers required per month (i.e., $W = \max_{k \in K}\{w_k\} = 7$)

The problem can be formulated on a directed graph $G = (N, A)$, where each node of N is defined as k^r, where $k \in K$, $r = 3$ if $k = 2, 9$ and $r = 0, 1, \ldots, W$ if $k \in K \setminus \{2, 9\}$. A node $k^r \in N$ represents the decision of deploying r workers at the end of month k. The arc set contains an arc (i^r, j^s) for each pair of nodes $i^r, j^s \in N$ satisfying the following conditions: $j = i + 1$, $r - 1 \leq s \leq r + 3$. The cost of each arc (i^r, j^s) is defined as follows $c_{i^r j^s} = \max\{s - r, 0\} \cdot 100 + \max\{r - s, 0\} \cdot 160 + |s - w_j| \cdot 200$. The personnel planning problem calls for finding a shortest path from node 2^3 to node 9^3 in this graph. A shortest path can be found with the Dijkstra's algorithm.

Exercise 4.5.

Modify the Dijkstra's algorithm so that it identifies a shortest directed path from each node $j \in N \setminus \{t\}$ to a given node $t \in N$.

The required changes are:

- we initialize the distance label of node t to zero instead of node s;
- we replace the vector of predecessors $pred(\cdot)$ with $next(\cdot)$, where $next(i)$ represents the next node visited after i in the shortest paths;
- at each iteration, once node i having the minimum distance among the temporary nodes is selected, we examine the incoming arcs $(j, i) \in A$ and update label $d(j)$ as $d(j) = d(i) + c_{ji}$ if $d(j) > d(i) + c_{ji}$.

Algorithm 2: Modified Dijkstra's Algorithm

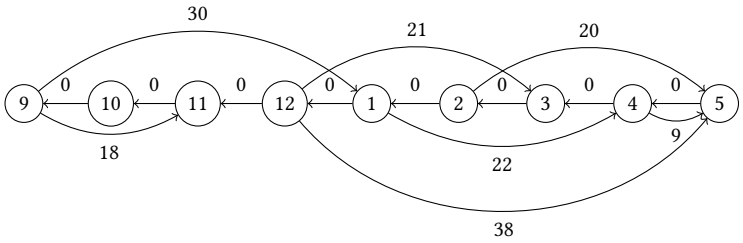
```
1  P ← ∅; T ← N;
2  d(i) ← ∞ for each i ∈ N;
3  d(t) ← 0 and next(t) ← 0;
4  while P ≠ N do
5      Let i ∈ T be a node for which d(i) = min{d(j) : j ∈ T};
6      P ← P ∪ {i}; T ← T \ {i};
7      for each (j, i) ∈ A do
8          if d(j) > d(i) + c_ji then
9              d(j) ← d(i) + c_ji;
10             next(j) ← i;
```

Exercise 4.6.

The following table illustrates some possible duties for the drivers of a bus company. We wish to ensure, at the lowest possible cost, that at least one driver is on duty for each hour of the planning period (9 a.m. to 5 p.m.). Formulate and solve this scheduling problem as a SPP.

Duty hours	9 - 1	9 - 11	12 - 3	12 - 5	2 - 5	1 - 4	4 - 5
Cost	30	18	21	38	20	22	9

We can construct a network $G = (N, A)$, where N contains a node for each hour from 9 a.m. to 5 p.m. (9 nodes in total). The arc set A consists of two sets of arcs $A = A^d \cup A^h$. A^d contains an arc between node s and node e for each duty that starts at time s and ends at time e. A^h contains an arc (h_{i+1}, h_i) between each pair of consecutive hours $i \in \{9, 10, \ldots, 3, 4\}$. The cost of each arc in A^d corresponds to the cost of the corresponding duty. All arcs in A^h has an associated zero cost.



A shortest path from the node corresponding to 9 a.m. to the node corresponding to 5 p.m. determines an optimal crew scheduling. The shortest path is 9 – 1 – 4 – 5; hence, the optimal set of duty hours is 9 – 1, 1 – 4, and 4 – 5, with a total cost of 61 units.