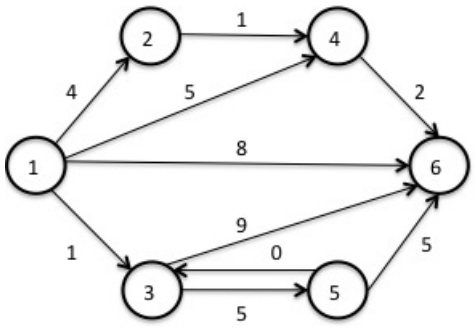
BGSE MSc in Data Science - Stochastic Models and Optimization
Instructor: Mihalis G. Markakis

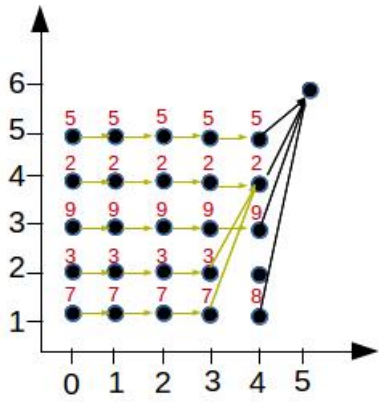
Problem Set 2 - Solutions

Problem 1 (Shortest Path via DP - [B05] Exercise 2.1)

Find a shortest path from each node to node 6 for the graph below by using the DP algorithm.

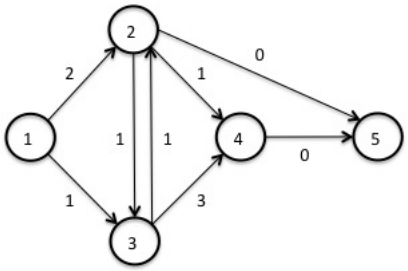


Solution:



Problem 2 (Shortest Path via Label Correcting Methods - [B05] Exercise 2.2)

Find a shortest path from node 1 to node 5 for the graph below by using both the Bellman-Ford and Dijkstra's methods.



Solution: *(a) The Bellman-Ford method.* The Bellman-Ford method uses a FIFO approach for extracting nodes from the OPEN set: the first node to enter the set is the first node to exit it. This approach is also known as breadth-first search. Using this approach, the iterations are as follows:

Iteration	Node Exiting OPEN	Nodes in OPEN	UPPER
1	-	1	∞
2	1	2(*2*), 3(*1*)	∞
3	2	3(*1*), 4(*3*)	2
4	3	4(*3*)	2
5	2	-	2

- **Iteration 1:** We start with the initial node 1 in OPEN. The value of UPPER is ∞
- **Iteration 2:** Node 1 exits OPEN. The children nodes 2 and 3 enter OPEN. The value of UPPER is still ∞ as we haven't reached the terminal node.
- **Iteration 3:** Node 2 exits OPEN. The children nodes 4 enters OPEN. Node 3 is a child of 2 but is already in OPEN. As we have reached the terminal node 5, UPPER can be updated to the length of the path so far, that is 2.
- **Iteration 4:** Node 3 exits OPEN, but its child node 4 is already in OPEN. As we haven't found a new path to node 5, the value of UPPER is not updated.
- **Iteration 5:** Node 4 has a label which is larger then UPPER so we discard it. Node 2 exits OPEN again, but this does not lead to updating UPPER as no shorter path is available.

As the OPEN set is empty, the algorithm terminates. It has found two equivalent shortest paths, that is: $1 \to 2 \to 5$ and $1 \to 3 \to 2 \to 5$ both of length 2.

(b) Dijkstra's method. With Dijkstra's method, the node that exits the OPEN set is that with the smallest label. Using this approach, the iterations are as follows:

- **Iteration 1:** We start with the initial node 1 in OPEN. The value of UPPER is ∞
- **Iteration 2:** Node 1 exits OPEN. The children nodes 2 and 3 enter OPEN. The value of UPPER is still ∞ as we haven't reached the terminal node.

Iteration	Node Exiting OPEN	Nodes in OPEN	UPPER
1	-	1	∞
2	1	2(*2*), 3(*1*)	∞
3	3	2(*2*), 4(*4*)	∞
4	2	4(*3*)	2
5		-	2

- **Iteration 3:** Node 3 exits OPEN because it has the smallest label. Node 2 is a child of 3, but it is already in OPEN and, moreover, its label does not require updating. The child node 4 enters OPEN with a label of 4. We haven't reached the terminal node yet, so we do not update UPPER.
- **Iteration 4:** Node 2 exists OPEN because it has the smallest label. As we have reached the terminal node, we can update UPPER to the value of 2. The label of node 4 can be updated to 3.
- **Iteration 5:** Both nodes in OPEN have labels which are higher than the current value of UPPER, so they can be discarded.

As the OPEN set is empty, the algorithm terminates. It has found one shortest path, that is $1 \to 3 \to 2 \to 5$ with length 2.

Notice that both algorithms find a shortest path of equal length, but the first approach identifies two equivalent paths because it explores the breadth of the tree before going down one branch. The second approach narrows down on the path with the nodes with the smallest level at each stage, thus it identifies one path but not the other.

Problem 3 (Clustering - [B05] Exercise 2.8)

We have a set of N objects, denoted $1, 2, \ldots, N$, which we want to group in clusters that consist of consecutive objects. For each cluster $i, i+1, \ldots, j$, there is an associated cost a_{ij}. We want to find a grouping of the objects in clusters such that the total cost is minimum. Formulate the problem as a shortest path problem, and write a DP algorithm for its solution.

Solution: We identify clusters with stages of the DP iteration: there can be at most N distinct clusters, or equivalently, exactly N distinct plus degenerate ones. Optimal clustering is equivalent to finding a shortest path from the artificial starting node 0 to the termination node N, in a graph where there is an arc of length $\alpha_{i+1,j}$ between every node $i \in \{0, \ldots, N-1\}$ and $j \in \{i+1, \ldots, N\}$. So, each arc (i, j) with $i < N$ corresponds to a cluster of nodes $i+1, i+2, \ldots, j$. On the other hand, the arc (N, N) corresponds to a degenerate cluster and has length zero. We have the following DP model:

State: the last node of the k^{th} cluster, denoted by x_k;

Control: the last node of the $(k+1)^{th}$ cluster, denoted by u_k;

Dynamics: $x_{k+1} = u_k$;

Cost: $g_k(x_k, u_k) = \alpha_{x_k+1,u_k}$, if $x_k \neq N$, and $g_k(N, N) = 0$;

Constraints: $U_k(x_k) = \{x_k + 1, \ldots, N\}$, if $x_k \neq N$, and $U_k(N) = \{N\}$.

The corresponding DP algorithm is as follows:

$$J_k(N) = 0, \qquad k \in \{1, \ldots, N\},$$

and

$$J_k(i) = \min_{j>i} \big\{\alpha_{i+1,j} + J_{k+1}(j)\big\}, \qquad i \neq N.$$

The optimal clustering has total cost $J_0(0)$.

Problem 4 (Path Bottleneck Problem - [B05] Exercise 2.15)

Consider the framework of the shortest path problem. For any path P, define the *bottleneck arc* of P as an arc that has maximum length over all arcs of P. We wish to find a path whose length of bottleneck arc is minimum, among all paths connecting the origin node to the destination node. Develop and justify an analog of the label correcting algorithm that solves this problem.

Solution: Let us reinterpret label d_i as the minimum bottleneck among the paths connecting the origin node and node i that have been considered thus far. The following variant of the label correcting algorithm solves the problem of finding a path with minimum bottleneck:

Step 1: Remove a node from OPEN, say node i, and for every child j of i execute Step 2;

Step 2: If $\max\{d_i, \alpha_{ij}\} < \min\{d_j, \text{UPPER}\}$, then set $d_j = \max\{d_i, \alpha_{ij}\}$ and i to be the parent of j. In addition, if $j \neq t$ put j in OPEN if it not already there, while if $j = t$ set UPPER $= \max\{d_i, \alpha_{ij}\}$.

Step 3: If OPEN is empty terminate; otherwise, go to Step 1.