

H: Homesick

Problem author: Jonas van der Schaaf



Problem: In an undirected unweighted simple graph, find a shortest walk from v_1 back to v_1 using at least one edge and without a *digon* (i.e., without a subwalk of the form u, v, u).

H: Homesick

Problem author: Jonas van der Schaaf



Problem: In an undirected unweighted simple graph, find a shortest walk from v_1 back to v_1 using at least one edge and without a *digon* (i.e., without a subwalk of the form u, v, u).

Idea: Assume there is a valid solution, and look at a node w on the path that is *furthest* from v_1 (i.e. the node with the greatest depth from v_1).

H: Homesick

Problem author: Jonas van der Schaaf



Problem: In an undirected unweighted simple graph, find a shortest walk from v_1 back to v_1 using at least one edge and without a *digon* (i.e., without a subwalk of the form u, v, u).

Idea: Assume there is a valid solution, and look at a node w on the path that is *furthest* from v_1 (i.e. the node with the greatest depth from v_1).

Observation: From this node, two paths to v_1 must exist that start out towards different neighbours of w .



Problem: In an undirected unweighted simple graph, find a shortest walk from v_1 back to v_1 using at least one edge and without a *digon* (i.e., without a subwalk of the form u, v, u).

Idea: Assume there is a valid solution, and look at a node w on the path that is *furthest* from v_1 (i.e. the node with the greatest depth from v_1).

Observation: From this node, two paths to v_1 must exist that start out towards different neighbours of w .

Observation 2: In the optimal solution, the sum of the lengths of these paths must be minimal.



Problem: In an undirected unweighted simple graph, find a shortest walk from v_1 back to v_1 using at least one edge and without a *digon* (i.e., without a subwalk of the form u, v, u).

Idea: Assume there is a valid solution, and look at a node w on the path that is *furthest* from v_1 (i.e. the node with the greatest depth from v_1).

Observation: From this node, two paths to v_1 must exist that start out towards different neighbours of w .

Observation 2: In the optimal solution, the sum of the lengths of these paths must be minimal.

Observation 3: The length of such a path is 1 plus the depth of the neighbour of w the path goes through.



Problem: In an undirected unweighted simple graph, find a shortest walk from v_1 back to v_1 using at least one edge and without a *digon* (i.e., without a subwalk of the form u, v, u).

Idea: Assume there is a valid solution, and look at a node w on the path that is *furthest* from v_1 (i.e. the node with the greatest depth from v_1).

Observation: From this node, two paths to v_1 must exist that start out towards different neighbours of w .

Observation 2: In the optimal solution, the sum of the lengths of these paths must be minimal.

Observation 3: The length of such a path is 1 plus the depth of the neighbour of w the path goes through.

Observation 4: These paths go to the two neighbours of w that are closest to v_1 .

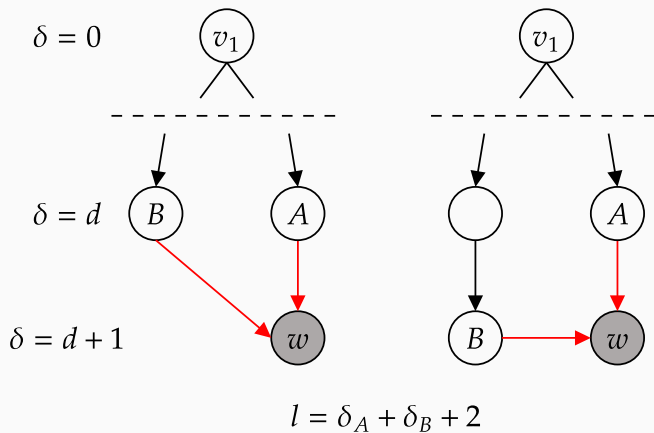


Figure 1: The two cases for optimal solutions given a furthest node.

H: Homesick

Problem author: Jonas van der Schaaf



Solution: Run a BFS and check the optimal path length for all possible deepest nodes:

H: Homesick

Problem author: Jonas van der Schaaf



Solution: Run a BFS and check the optimal path length for all possible deepest nodes:
Mark the depth of each node.

H: Homesick

Problem author: Jonas van der Schaaf



Solution: Run a BFS and check the optimal path length for all possible deepest nodes:

Mark the depth of each node.

When encountering a node for the second time, the path length is the depth of the node plus the path length of the second path to this node.

H: Homesick

Problem author: Jonas van der Schaaf



Solution: Run a BFS and check the optimal path length for all possible deepest nodes:

Mark the depth of each node.

When encountering a node for the second time, the path length is the depth of the node plus the path length of the second path to this node.

Pick the optimal deepest node (if one exists) and reconstruct the path.



Solution: Run a BFS and check the optimal path length for all possible deepest nodes:

Mark the depth of each node.

When encountering a node for the second time, the path length is the depth of the node plus the path length of the second path to this node.

Pick the optimal deepest node (if one exists) and reconstruct the path.

Running time: $\mathcal{O}(n + m)$

H: Homesick

Problem author: Jonas van der Schaaf



Solution: Run a BFS and check the optimal path length for all possible deepest nodes:

Mark the depth of each node.

When encountering a node for the second time, the path length is the depth of the node plus the path length of the second path to this node.

Pick the optimal deepest node (if one exists) and reconstruct the path.

Running time: $\mathcal{O}(n + m)$

Statistics: 123 submissions, 24 accepted, 43 unknown