Let $G = (V, E, w)$ be a directed and weighted graph with positive edge weights $w(e) > 0$ for each edge $e \in E$. For a pair of vertices $u, v \in V$, there may be *multiple* shortest paths from u to v. Let $\Pi_{u,v}$ denote all such paths. In other words, for a pair of vertices u and v, $\Pi_{u,v}$ is the set of all shortest paths from u to v.

A vertex x is called *useful* if x lies on *any* path in $\Pi_{u,v}$. Given the graph G and a pair of vertices $u, v \in V$, describe an $O(m \log n)$ algorithm to return all useful vertices.

Solution. Applying Dijkstra's algorithm on G with source u will associate each vertex $v \in V$ with the set of possible predecessors of v in the shortest paths from u to v. In other words, Dijkstra's algorithm gives us the function $\mathrm{pred} : V \to \mathcal{P}(V)$, where pred is defined such that for all $v \in V$, $\mathrm{pred}(v) = \{x \in V : (\exists p \in \Pi_{u,v} \text{ such that } (x, v) = p.\textit{back})\}$. We can return a reference to a predecessor set of a vertex from this function in $O(1)$ time.

Let $B = \{(a, b) \in V \times V : b \in \mathrm{pred}(a)\}$, representing the set of all edges which exist in shortest paths, but reversed. Let $A = (V, B)$ be a directed and unweighted graph. Perform a BFS on A from the starting vertex v, using the pred function as an adjacency list. Insert all processed vertices into a set and return the set.

This algorithm is correct because if a is the predecessor of v in a shortest path from u to v, then all useful vertices from u to a are also useful vertices from u to v.

The Dijkstra's algorithm step is $O(m \log n)$. The BFS step will take $O(m)$ time. Therefore the whole algorithm together is $O(m \log n + m) = O(m \log n)$.