The approximation solution is a greedy algorithm that finds the longest path in a graph.
The algorithm works like this:

1. Start at a random unvisited vertex.
   - Remove selected vertex from the set of unvisited vertices.
   - If no vertex is available, stop.
2. Search all neighbors of the current vertex. The neighbor with the highest edge weight becomes the current vertex.
   - Upon selection, mark the neighbor as visited and add its connecting edge weight to the current path weight.
   - Previously visited vertices are ignored, including the start vertex.
   - If no neighbor is available, stop.
3. Repeat step 2 until the current path cannot be extended further.
4. Validate the computed path. If it is valid and longer than the current best path, update the best path.
5. Repeat step 1 until all vertices have been chosen as the start vertex or the time limit is exceeded.

The program terminates in two cases:

1. All vertices have been selected as the start vertex.
2. The time limit has been exceeded (Default time limit is 30 seconds).

Analytical Runtime Analysis:
- n is the number of vertices in the graph
- m is the number of edges in the graph
- k is the number of iterations before the time limit is reached

1. Graph Construction: O(m)
   - Each edge is appended to the adjacency list

2. Greedy Search: O(n * m)
   - In the worst case, the graph is a complete graph, meaning that for each vertex, all edges must be visited to find the biggest weight.

3. Validation: O(n * m)
   - Iterates through all vertices in the provided path
   - In the worst case, the graph is a complete graph, so when traversing a path, each vertex could have to visit every edge in the graph

4. Main Loop (Total Complexity): O(k * (n * m))
   - Involves choosing the start, searching for paths, and validating paths
   - Each iteration of the main loop is O(n * m)
   - If k iterations are performed, the total complexity is O(k * (n * m))