Solution. Begin by transforming the arcs into linear intervals by considering their angular range. For any interval crossing the 0-degree point, split those into two arcs, keeping track of the fact that they are actually the same arc. Next, we sort the intervals by their ending points. If two intervals have the same ending point, order them by their starting point. This process will be done in $O(n \log n)$ time.

We now initialise a Dynamic Programming table, say D, with $D[0] = 0$. This table will represent the size of the largest set of non-intersecting arcs from before the index. For each interval i from 1 to n, we must find the rightmost interval $j < i$ that does not intersect interval i by iterating back from $i-1$. This check will take $O(n)$. Then, we update the table as $D[i] = \max\{D[i-1], D[j]+1\}$. This way, we either take the current interval and the best solution up to j, or not taking the current interval and using the best solution up to $i-1$.

By the time we reach the end, $D[n]$ will represent the size of the largest set of non-intersecting arcs. If any intervals were split in the intial transformation, we would manually ensure that they are included/excluded together in the final value. This process will run n operations of $O(n)$ time, making it $O(n^2)$.