Kevin Lopez

CIS473

05/09/2023

Final Paper

Time Complexity

Time complexity is a way of measuring how much time will an algorithm take to solve a problem. They are described in polynomial time with either: n, nlogn,n^2 and so forth. It is dominated by the largest term of expression. It is computed by the length of the string representing input and is divided by worst-case running time and average-case running time.

The worst-case time complexity of most sorting algorithms are n^2 because they require filtering through a list:n, in n time but merge sort is nlogn because it cuts each list in half when filtering each time.

NP Completeness

This involves certain problems whose complexities related to an entire class. It shows that if a polynomial time algorithm exists for these problems, all problems would be polynomial time solvable. This summarized up shows that if you can solve a problem with a solution to another problem, then they are both NP-complete. In addition, For a language to be NP-complete, it must be NP and reducible by another language A in NP.

An example of this is the Hamiltonian cycle. It is a cycle in an undirected graph(G=(V,E) that traverses each vertex exactly once.

To prove this is NP, we will have to create a polynomial time verification of the solution to a problem. To do this, we can check if all the vertices of the graph belong to the graph and if all the vertices belonging to them are adjacent using an O(V+E) time graph. This is also NP hard by creating a new graph from the Hamiltonian path problem where it is now G’=(V’,E’). In V’ we add vertices to the original graph and add one new vertex that connects to each part of the graph. Then for the edges, We add the original graph edges and add new edges to the newly added vertex. Then we run the algorithm because now we know G’ will have a closed cycle traversing all vertices once.