Assignment No. 2

1 TITLE

Analysis of algorithms and mathematical modelling.

2 AIM

Improving Human Computer Interaction with Machine Emotional Intelligence using NAO robot.

3 OBJECTIVE

3.1 To learn analysis of algorithm

3.2 To learn algorithmic strategies

3.3 To draw P and NP for project

3.4 To learn how to draw mathematical model for different models of the project

4 NEED FOR ANALYSIS TO BE DONE

Informally an algorithm is any well-defined computational procedure that takes some value or a set of values as input and produces some value, or set of values as output. An algorithm is thus a sequence of computational steps that transform the input into output. To analyse an algorithm is to determine the amount of resources (such as time and storage) necessary to execute it. In order to choose the best algorithm for a particular task, we need to be able to judge how long a particular solution will take to run; or how long two solutions will take to run and choose the better of the two. We don't need to know how many minutes and seconds they will take, but we

need some way to compare algorithms against one another. This is why we need to analyse an algorithm.

5 TIME COMPLEXITY

P, NP, NP-complete, NP-hard

5.1 P:

If the running time is some polynomial function of the size of input, for instance if the algorithm runs in linear time or quadratic time or cubic time, then we say that algorithm runs in polynomial time and the problem solves in class P.

5.2 NP:

Now, there are lot of programs that don't run in polynomial time on regular computer, but do run in polynomial time on nondeterministic Turing machine. These programs solve the problems in NP, which stands for nondeterministic polynomial time. A nondeterministic Turing machine can do everything a regular computer can do and more. This means there is not necessarily a polynomial time way to find a solution, but once you have a solution it takes only polynomial time to verify that it is correct.

5.3 NP-complete:

NP is a complexity class which represents the set of all problems X for which it is possible to reduce any other NP problem Y to X in polynomial time. Intuitively this means that we can solve Y quickly if we know how to solve X quickly. Precisely, Y is reducible to X, if there is a polynomial time algorithm f to transform instances y of Y to instances x = f(y) of Xin polynomial time, with the property that the answer to y is yes, if and only if the answer to f(y) is yes.

5.4 NP-hard:

Intuitively, these are the problems that are even harder than the NP-complete problems. Note that NP-hard problems do not have to be in NP, and they do not have to be decision problems. The precise definition here is that a problem X is NP-hard, if there is an NP-complete problem Y, such that Y is reducible to X in polynomial time. But since any NP-complete

problem can be reduced to any other NP-complete problem in polynomial time, all NP-complete problems can be reduced to any NP-hard problem in polynomial time. Then, if there is a solution to one NP-hard problem in polynomial time, there is a solution to all NP problems in polynomial time.

6 TODO ALGORITHM TYPE

7 TODO FEASIBILITY STUDY

7.1 Technical Feasibility

7.2 Economic Feasibility

7.3 Time Feasibility

7.4 Privacy Feasibility

8 TODO MATHEMATICAL MODEL