AN ALGORITHM TO FIND THE DIFFICULTY LEVEL OF A QUESTION

INTRODUCTION :

In recent times the rate of development of online courses or classes have increased drastically due to the global pandemic. In relation to that, this project is all about finding the difficulty level of any particular question using **statistical formulas** and **analysis by difficulty index value**. By analyzing the difficulty of a question,we can also **conclude the skills of a particular student** which can be used for improving his skills thereby knowing the level of a student. Our primary aim is to find the difficulty level of a Multiple choice question / Fill in the blanks / Match the following / Coding question based on the details provided for each of the question.A **low p-value** (difficulty index value) means that a **low percentage of participants answered the question correctly**. The question can be difficult or can contain incorrect information. If the questions in a test cover the same theme and have the same difficulty level. For example, Hackerrank has around 5 levels of difficulties like Easy, Intermediate, Hard, Expert and Advanced which will more accurately determine the level of the question.

LANGUAGE USED :

- Python 3.7

GITHUB LINK:

- https://github.com/kaviyasribalakrishnan/FindDifficulty.git

ALGORITHM / PSEUDOCODE:

1. Get input as a file containing required data as follows,
    a. Type of the question (say choice)
    b. Difficulty level of the problem (say difficulty)
    c. Number of students takes the question (say n)
    d. Time taken by each student (say time)
    e. Number of times answer changed by each student (say change answer)
    f. Feedback given by each student (say feedback)
    g. Total number of students answered correctly (say totcorrect)
    h. Total number of students answered partially correct (say totpartial)
    i. Total number of students answered it wrongly (say totwrong)
    j. Maximum marks for the particular question (say maxmarks)

2. The solution can be divided into two parts, first part is achieved by processing the data based on the **performance of students metrics** like time taken, change answer, feedback and the second part is achieved by processing the data based on the **difficulty index** calculated.

3. The maximum time taken, attempts to change answer for each question is manually set according to the level of a question. If the performance metrics of a student is **compared** with the assigned values and the **level is found**. And this will be the solution for the part 1.

4. By formulating the values of **students who got correct** answer for the question versus the **number of students** who took the test, we can find the **difficulty index**. The solution for the part 2 is found by comparing the difficulty index with the assigned values.

5. By combining the solution from the both parts of the question we can arrive at the final solution defining the level of the question as Easy / Medium / Hard.

This solution can be further improved by employing certain Machine Learning Algorithms like e-learning algorithm, Regression algorithm. By implementing the k-means algorithm the solution will yield a 20 - 25% accuracy. We can group the data into 3 different groups like, students who got fully correct, partially correct and wrong answers.

PSEUDOCODE :

```
Function part1mcq (time,attempts,feedback,n) {
        i=0
        for i to n {
                if (time[i] > 90 ) then add 75 points (hard)
                else if (time[i] >60 ) then add 50 points (medium)
                else add 25 points (easy)

                if(attempts[i] > 4 ) then 75 points (hard)
                else if(attempts[i] >2 ) then 50 points (medium )
                else 25 points (easy)

                if(feedback[i] == easy) then add 25 points (easy)
                else if(feedback[i] == medium ) then add 50 points (medium)
                else if (feedback[i] == hard) then add 75 points (hard)

                take average for the added values ie.
                total = total/3
        }
    return the total/n
}
```

```
Function part1coding (time_in_minutes,times_compiled,feedback,n)
{           i=0
            For i to n {
                  if (time[i] > 45 ) then add 75 points (hard)
                  else if (time[i] >15 ) then add 50 points (medium)
                  else add 25 points (easy)

                  if(attempts[i] > 10 ) then 75 points (hard)
                  else if(attempts[i] >5 ) then 50 points (medium )
                  else 25 points (easy)

                  if(feedback[i] == easy) then add 25 points (easy)
                  else if(feedback[i] == medium ) then add 50 points
             (medium)
                  else if (feedback[i] == hard) then add 75 points (hard)
                  take average for the added values ie.
                  total = total/3
             }
        return the total/n
    }


Function Part2 ( ncorrect, nwrong, npartial, n ) {
        ans = (ncorrect+npartial)/n
        if( ans >=0.75) then return 25 (easy)
        else if(ans <0.25) then return 75 (hard)
        else    return 50
}

Function PrintDiff ( rows , n ){
        total=0.0
        for x in rows {
              Add x to total
           }
         total=total/n
         if(total<100.0)    then print("Easy Question")
         else if(total<200.0)    then print("Medium Question")
         else print("Hard Question")
  }
```

PROOF :

1. DIFFICULTY INDEX :

The Difficulty index is the proportion or probability that candidates, or students, will answer a test item correctly. Generally, more difficult items have a lower percentage, or P-value.The formula for finding difficulty index is the number of students who answer a question correctly (c) divided by the total number of students who answered the question (s).

$$\text{Difficulty index} = \frac{\text{Total number of students who answered correctly (c)}}{\text{Total number of students who answered the question(s)}}$$

Comparing with the range of values, we can assign particular points for difficulty range.

S.No	Range	Value
1.	>=0.75	Easy (25 points)
2.	<0.25	Hard (75 points)
3.	Otherwise	Medium (50 points)

The average time taken for an easy / medium / hard coding question in tests are shown below,

S.No	Time taken (in minutes)	Difficulty
1.	<=15 minutes	Easy (25 points)
2.	<=30 minutes	Medium (50 points)
3.	>45 minutes	Hard (75 points)

Maximum time limit for Multiple choice / fill in the blanks / match the following will probably have same time limit or it will differ slightly greater than each of others. Here we give the time taken by each student will be given in seconds to accurately derive values or points.

We will compare the feedback given by various students and find the average mean value for all the feedback given by total number of students who attended the question. The number of times answer changed in case of Mcq / fillups / match the following and number of times compiled in case of a coding question.On finding the average mean for the both solutions of part 1 and part 2 we will get the final solution.

For Example :

choice	difficulty	n	Time (in sec)	attempts	feedback	Total correct	Total wrong	Total partial	Max marks
Mcq	Easy	5	12,32 16, 24,18	1,2,1,2,3	Easy,easy,medium,easy,easy	4	1	0	5

Difficulty index = 4/5=0.8 ,value >=0.75 -----> 25 points (easy)
Initially assigned difficulty = easy ---> 25 points (easy)

Avg value :

i. 12 --> 25 points (easy), 1 -- >25 points (easy) , easy --->25 points (easy)

ii. 32 --> 25 points (easy), 2 --> 25 points (easy) , easy --->25 points (easy)

Similarly, the average will be found by using the comparison with the given values.

Hence this question will produce the result as “ Easy ”

PERFORMANCE METRICS :

Program metrics are standard measurements for monitoring, evaluating and bench marking an ongoing program. These include metrics for quality, results, effectiveness and projects. Performance metrics, also known as KPIs (Key Performance Indicators), measure how a process performs and reflect the accomplishment of individual goals.

Performance metrics generate data, bringing more efficiency and effectiveness to the decision-making process.

Example :

1. 100 Question 100 attempts :
   Run time - 0.000008s

2. 1 Question and 100 attempts :
   Run time - 0.000012s

3. 100 Questions 1 attempt :
   Run time - 0.000013s

4. 10 Questions 100 attempts :
   Run time - 0.000013s

Since python runs program at a faster rate comparing to other programming languages like C , C++ , Java. It will produce at most same run time for the attempts like 1million questions and 1 million attempts, etc. It provides a very smaller run time compared to other programming languages.

FURTHER IMPROVEMENTS :

In order to get 90 - 100 % accuracy, it can be further enhanced with ML algorithms and model that are trained using the data set and which will produce accurate level of difficulty of any particular question provided with required input data.This Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion.