**CONCEPT**

Suppose we are given a set of data where we need to find a requested element (searching basically), we normally implement searching techniques (algorithms) using only a single searching **agent**. What if we could implement multiple **agents** to search simultaneously in different parts of the given set of data?

**Example**: A tennis ball is lost on a playground, and then instead of searching different locations by a single person, we would normally divide our team to look for the tennis ball in different directions (locations). This ‘divide and search’ technique is **efficient** and yet effort-less.

**Explanation:** consider the data set of size 9.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | **9** | **17** | **36** | **39** | **74** | **88** | **97** | **112** |

**Agent** (1) **Agent** (2) **Agent** (3)

Let us have **N** number of data elements (N=9) in a set, where we have assigned **M** number of **Agents** (M=3) to search for a particular element in a sub-set (part) of the given set. Thus each **Agent** is having **N**/**M** numbers of data to search concurrently for the required element.

Now, if we program the best possible searching algorithm (till date) to each of the Agents then we could significantly reduce the time complexity of searching.

**Sample Test:** Let us apply the **Binary search** algorithm to the above mentioned data set.

The worst time complexity is **O(log N)** i.e. O(log 9). If we use the above concept we divide the data set into **M** number of sub-sets (M=3) and assign each agent to look concurrently for the element using Binary search then the time complexity for each agent becomes **O(log(N/M))** i.e. O(log 3),thus at O(log(N/M)) we could search **N** number of data in a given set.

Let us check for **1,000,000** data in a set (N=1,000,000).

|  |  |  |
| --- | --- | --- |
| Number of **Agents** searching (M) | Data for each agent(**N/M**) | Time Complexity **O (log (N**/**M)) ,** approximate value |
| **1** | **1,000,000** | **19.931** |
| **2** | **500,000** | **18.931** |
| **5** | **200,000** | **17.609** |
| **10** | **100,000** | **16.609** |
| **100** | **10,000** | **13.287** |
| **1000** | **1000** | **9.965** |

**Ideally: If M=N,** then we get time complexity of **O(1).**

“***I*** *don’t know whether this concept is feasible enough to implement in real life applications, please do share your thoughts regarding this concept*”

**Shiladitya Goswami**