Searching

Searching is a process of retrieving or locating a record with a particular key value.

Sequential / Linear Search:

This is the simplest form of searching. It can be applied for sequential storage structures like files, arrays or linked lists. In this method, the searching begins from the first record. The required key value is compared with the record key. Searching continues sequentially till the record with a matching key value is found or if the table ends.
Although not required, we usually consider the list to be unsorted when doing a sequential search, because there are other algorithms that perform better on sorted lists. Sequential search looks at elements, one at a time, from the first in the list until a match for the target is found.

Advantages :

- It is a very simple method.
- It does not require the data to be ordered.

Disadvantage :

- If n is very large, this method is very inefficient and slow

Analysis of Linear Search :
Best Case : Record found at first position itself, only 1 comparison is required.
Worst Case : Record not present in the list, n+1 comparisons are required.
Average Case : For all keys Ki ($1 <= i < n$) the total number of comparisons for successful searches are : $1 + 2 + \ldots + n = (n + 1) / 2 = O(n)$
Hence the time complexity of this method is O(n).

<u>Binary Search</u>
This is a very efficient searching method used for linear / sequential data.
In this search, Data has to be in the sorted order either ascending or descending. Sorting has to be done based on the key values.
In this method, the required key is compared with the key of the middle record. If a match is found, the search terminates. If the key is less than the record key, the search proceeds in the left half of the table. If the key is greater than record key, search proceeds in the same way in the right half of the table. The process continues till no more partitions are possible. Thus every time a match is not found, the remaining table size to be searched reduces to half.

If we compare the target with the element that is in the middle of a sorted list, we have three possible results: the target matches, the target is less than the element, or the target is greater than the element. In the first and best case, we are done. In the other two cases, we learn that half of the list can be eliminated from consideration.
When the target is less than the middle element, we know that if the target is in this ordered list, it must be in the list before the middle element. When the target is greater than the middle element, we know that if the target is in this ordered list, it must be in the list after the middle element. These facts allow this one comparison to eliminate one-half of the list from consideration. As the process continues, we will eliminate from

consideration, one-half of what is left of the list with each comparison. This technique is called as binary search.

Advantage :

- Time complexity is O(logn) which is very efficient.

Disadvantage :

- Data has to be in sorted manner.

Analysis:

After 0 comaparisons → Remaining file size = n
After 1 comaparisons → Remaining file size = n / 2 $= n / 2^1$
After 2 comaparisons → Remaining file size = n / 4 $= n / 2^2$
After 3 comaparisons → Remaining file size = n / 8 $= n / 2^3$
……
After k comaparisons → Remaining file size = $n / 2^k$

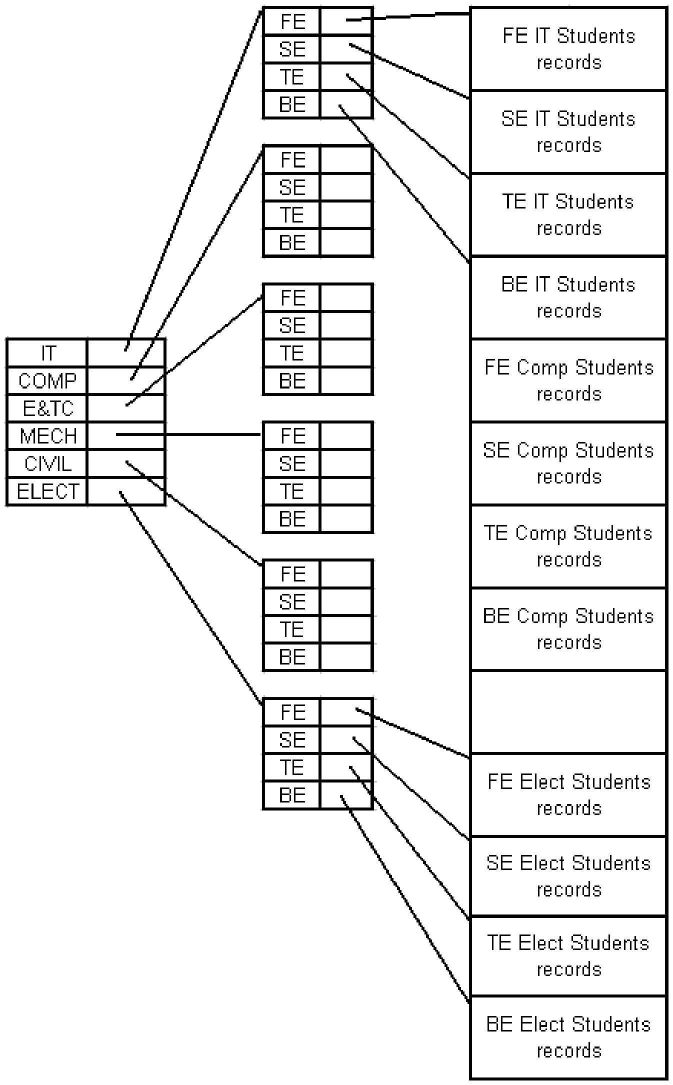
The process terminates when no more partitions are possible i.e. remaining file size = 1
$n / 2^k = 1$
$k = \log_2 n$
Thus, the time complexity is $O(\log_2 n)$. which is very efficient.

<u>Indexed Sequential Search:</u>

If we have a data of all students in the college, then we will arrange it as per years, i.e. FE, SE, TE and BE and then on branch and then alphabetically. Given only the name, for each year and branch, we will have to perform binary search.

- If there are 6 branches and each class contains 60 students, then number of comparisons required will be ($4 * 6 * \log_2 60$). Any additional information will always reduce the number of comparisons.
- If we know the year, then the number of comparisons required will be ($6 * \log_2 60$)
- If we know the branch, then the number of comparisons required will be ($4 * \log_2 60$)
- If we know both year and the branch, then the number of comparisons required will be ($\log_2 60$). **The year and branch are the additional keys used for searching.**

The only assumption made in the above discussion is that the strength of the class is uniform, i.e. 60. If it is varying, then the key year will inform the range of positions where the data of that year is stored. The information about the branch will further reduce this range to the class strength. This technique is known as indexing. In Index Sequential search, the data will inform us about the range where it could be present. In that range the search could be sequential.


IT
COMP
E&TC
MECH
CIVIL
ELECT
FE
SE
TE
BE
FE
SE
TE
BE
FE
SE
TE
BE
FE
SE
TE
BE
FE
SE
TE
BE
FE
SE
TE
BE
FE IT Students records
SE IT Students records
TE IT Students records
BE IT Students records
FE Comp Students records
SE Comp Students records
TE Comp Students records
BE Comp Students records
FE Elect Students records
SE Elect Students records
TE Elect Students records
BE Elect Students records