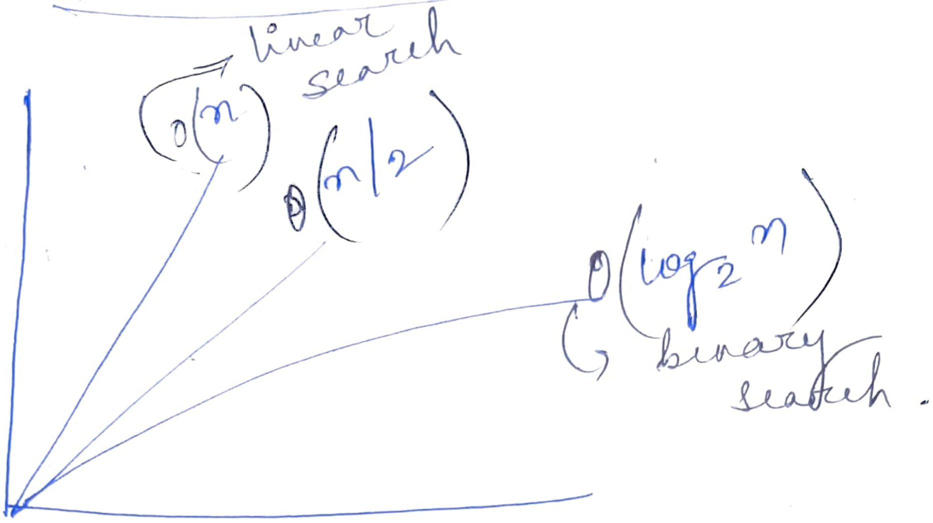
03/07/20

Lecture - 3 Algorithms



where n is the no. of steps.

← Computer ~~sciés~~ scientists determine algorithms or how well they are by a terminology known as Big O.

→ Big O means on the order of.

Efficieny of algorithm is determined with the help of Big O. It is just an approximation of how fast or slow The code is.

→ Running time is how much time it takes for your program or your algorithm to run. How many steps or seconds etc.

~~Linear Search $O(n)$~~

Running time of some algorithms

Better in terms of time req.

$O(n^2)$ — Bubble sort, selection sort.

$O(n \log n)$ merge sort

$O(n)$. — Linear search.

$O(\log n)$ — Binary search.

$O(1)$

⇒ Big O is essentially an upper bound on how much time an algorithm might take. Usually refers to the worst cases.

where as,

Ω is referred to the best cases, opposite of BigO. It refers to the lower bound on the running time.

$\Omega(n^2)$ — bubble sort, Selection sort

$\Omega(n \log n)$ — merge sort.

$\Omega(n)$ — bubble sort

$\Omega(\log n)$

$\Omega(1)$ — Linear search, Binary search

$\Omega(n)$ & $O(n)$

- counting any thing.
- It takes the same no. of step in the best case & worst case.

- One should look for the worst case.

→ In C, strings are arrays. So, can't compare to other string using ==. Use 'strcmp'. It returns 0 if two strings are the same.

→ We can define custom data types in C. using typedef.

→ Struct is a container where we can put multiple data types.

Bubble Sort Pseudocode

```
Repeat n-1 times
    for i from 0 to n-2
        if ith & i+1'th element out
                          of order
            swap them
```

running time

$(n-1)(n-1)$

$= n^2 - 2n + 1$

$= O(n^2)$

Pseudocode of Selection sort

```
n times ←  for i from 0 to n-1
n times ←      find smallest item between ith
                    item & last item
               swap smallest item with ith
                    item.
```

$$n + (n-1) + (n-2) + \dots + 1$$

$$= \frac{n(n+1)}{2} = \frac{n^2+1}{2} = \frac{n^2}{2} + \frac{1}{2}$$

$$= O(n^2)$$

Improved bubble sort

```
Repeat until no swaps
    for i from 0 to n-2
        if ith & i'th elements out of
        order
            swap them.
```

→

$\Omega(n)$

→ Merge Sort divides the problem in half each time, so running time is $O(\log n)$ + n → steps to look at each element once.

Θ Notation

To describe the running times of algoorthms if the upper bound and the lower bound is the same..

Ex – Merge sort $\Theta(n \log n)$

Selection sort $\Theta(n^2)$

2