Mathematical Analysis Linear Search

```
int linSearch(int a[], int n, int val) {
    for(int i=0; i<n; i++){
        if(val==a[i]) return i;
    }
    return -1;
}
```

1 operation Per n

$$T(n) = \sum_{i=1}^{n} 1 = n = O(n)$$

Mathematical Analysis of Binary Search

```
int binSrch(int a[], int n, int val){

    int lowEnd = 0;
    int highEnd = n-1;

    do {
        int middle = (highEnd + lowEnd) / 2;
        if(val == a[middle]) return middle;
        else if(val > a[middle]) lowEnd = middle + 1;
        else highEnd = middle - 1;
    }while(lowEnd <= highEnd);

    return -1;
}
```

Each loop the search size is cut in half.

n = Search Size
Op = operations

n is divided by 2 for each operation until the search is over when $n = 1$

$$n\left(\frac{1}{2}\right)^{op} = 1 \qquad \frac{n}{2^{op}} = 1 \qquad n = 2^{op}$$

Solve for Op

$$n = 2^{op}$$

$$\log_2(n) = \log_2(2^{op})$$

$$\log_2(n) = op$$

Upper bound $op = \log_2(n) = \boxed{O(\log n)}$