Data Structure & Algorithms

Nilesh Ghule

Course Introduction

- Data Structure and Algorithms
  - Data Structures: Linked list, Stack, Queue, Binary search tree, Heap, Graph.
  - Algorithms: Sorting, Searching, Stack, Queue & Linked list applications, Graph algorithms.
- Course Goals
  - Implement each DS & Algorithms from scratch.
  - Understand complexity of algorithms.
- Course Schedules
  - 5th Sep 2021 to 20th Sep 2020*
  - Mon-Fri: Lecture – 9:00 AM to 1:30 PM
  - Mon-Fri: Lab – 2:30 PM to 3:30 PM
- Resource sharing
  - https://gitlab.com/sunbeam-modular/dsa-04
  - Recorded videos will be available for 7 days. http://students.sunbeamapps.org
- Course Format
  - Participants are encouraged to code alongside (copy code from code-sharing utility in student portal).
  - Post your queries in chat box (on logical end of each topic).
  - Practice assignments will be shared. They are optional. If any doubts, share on WA group (possibly with screenshot). Faculty members or peers can help.
- Programming language
  - DS & Algorithms are language independent.
  - Classroom coding will be in Java (use IDE of your choice).
  - Will share C++/Python codes at the end of session.
  - Language pre-requisites ?

Course Pre-requisites

Java

- Language Funda
- Methods
- Class & Object
- static members
- Arrays
- Collections
  - ArrayList<>
  - Stack<>, Queue<>

Python

- Language Funda
- Functions
- Class & Object
- Collections

C++

- Language Funda
- Functions
- Class & Object
- Friend class
- Arrays
- Pointers

C

- Language Funda
- Functions
- Structures
- Arrays
- Pointers

Data Structure

- Data Structure
  - Organizing data in memory
  - Processing the data
- Common data structures
  - Array
  - Linked List
  - Stack
  - Queue
  - Hash Table
- Advanced data structures
  - Tree
  - Heap
  - Graph

Data Structure

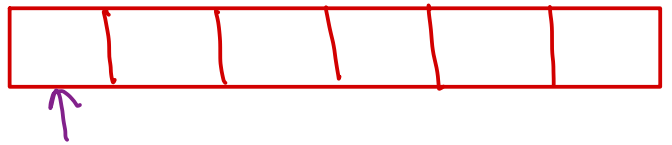
- Data Structure
  - Organizing data in memory
  - Processing the data
- Common data structures
  - Array
  - Linked List
  - Stack
  - Queue
- Advanced data structures
  - Tree
  - Heap
  - Graph

- Asymptotic analysis
  - It is not exact analysis
  - Big' O notation

Exact analysis
① How much time?
② How many bytes?
* Arch / machine dependent

- Space complexity
  - Unit space to store the data (Input space) and additional space to process the data (Auxiliary space).
  - O(1), O(n), O(n^2)

S ∝ n
+ input space: O(n)
aux space: S = k
O(1)



- Time complexity
  - Unit time required to complete any algorithm.
  - Approximate measure of time required to complete any algorithm.
  - Depends on loops in the algorithm.
  - O(n^3), O(n^2), O(n log n), O(n), O(log n), O(1)

T ∝ n

Time complexity

- Write a program to calculate factorial of given number.

$T \propto n$

$O(n)$

```
r = 1;
for (i=1; i<=n; i++)
    r = r * i;
print (r);
```

- Print 2-D matrix of n x n.

itrs = n * n

$T \propto n^2$

$O(n^2)$

```
for(i=0; i<n; i++){
    for(j=0; j<n; j++){
        print(mat[i][j]);
```

3

3

- Print given number into binary.

$2^{itr} = n$

$itr \log 2 = \log n$

$itr = \frac{\log n}{\log 2}$

$T \propto \frac{\log n}{\log 2}$

$T \propto \log n$

$O(\log n)$

$(10)_{10} = (1010)_2$

10 / 2 = 5 — 0

5 / 2 = 2 — 1

2 / 2 = 1 — 0

1 / 2 = 0 — 1

```
while(n > 0){
    print(n % 2);
    n = n / 2;
```

3

- Print table of given number.

```
for (i=1; i<=10; i++)
    print( n * i);
```

$T = k$

$O(1)$

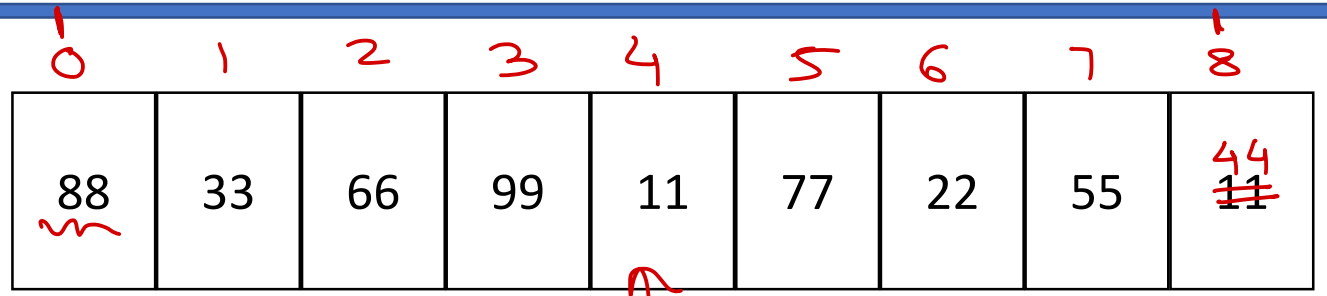
Linear Search

C → strchr()

array of 9 elements

- Find a number in a list of given numbers (random order).

key = 90




```
int LinearSearch( arr, n, key){
    for(i=0; i<n; i++) {
        if (arr[i] == key)
            return i;
    }
    return -1;
}
```

- Time complexity
  - Worst case $T \propto n$ $O(n)$
  - Best case $T = K$ $O(1)$
  - Average case $T \propto \frac{n}{2}$ $T \propto n$ $O(n)$

Binary Search

Key = 30

```
left = 0;
right = n-1;
while(left <= right) {
    mid = (left + right)/2;
    if(key == arr[mid])
        return mid;
    if(key < arr[mid])
        right = mid-1;
    else
        left = mid+1;
}
return -1;
```

0	1	2	3	4	5	6	7	8
11	22	33	44	55	66	77	88	99

R L

m

Best case (middle)

O(1)

$2^{itr} \simeq n$

$itr \log 2 = \log n$

$itr = \frac{\log n}{\log 2}$

$T \propto \frac{\log n}{\log 2}$

$T \propto \log n$

$O(\log n)$ ← avg/worst case

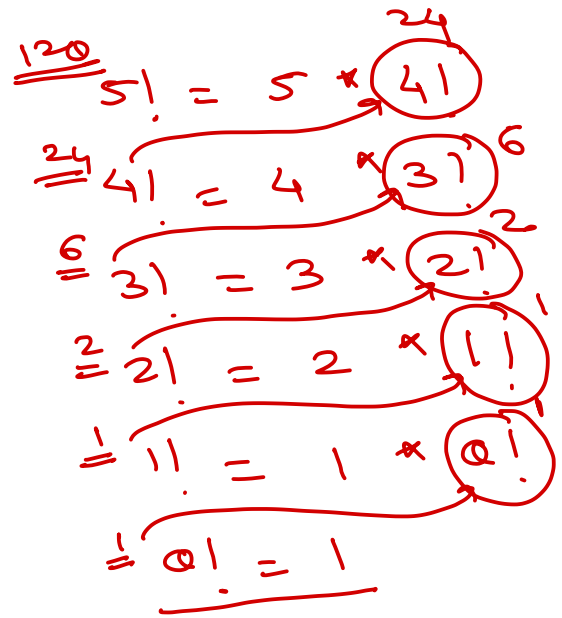
Recursion

- Function calling itself is called as recursive function.
- To write recursive function consider
  - Explain process/formula in terms of itself
  - Decide the end/terminating condition
- Examples:
  - $n! = n * (n-1)!$ $\quad 0! = 1$
  - $x^y = X * x^{y-1}$ $\quad x^0 = 1$
  - $T_n = T_{n-1} + T_{n-2}$ $\quad T_1 = T_2 = 1$
  - factors(n) = 1st prime factor of n * factors(n)

- On each function call, function activation record or stack frame will be created on stack.

```
int fact(int n) {
  int r;
  if(n==0)
    return 1;
  r = n * fact(n-1);
  return r;
}

res=fact(5);
```



Recursion

```
int fact(int n) {
   int r;
   if(n == 0)
      return 1;
   r = n * fact(n-1);
   return r;
}
```

```
int fact(int n) {
   int r;
   if(n == 0)
      return 1;
   r = n * fact(n-1);
   return r;
}
```

```
int fact(int n) {
   int r;
   if(n == 0)
      return 1;
   r = n * fact(n-1);
   return r;
}
```

```
int fact(int n) {
   int r;
   if(n == 0)
      return 1;
   r = n * fact(n-1);
   return r;
}
```

```
int fact(int n) {
   int r;
   if(n == 0)
      return 1;
   r = n * fact(n-1);
   return r;
}
```

```
int fact(int n) {
   int r;
   if(n == 0)
      return 1;
   r = n * fact(n-1);
   return r;
}
```

```
int main() {
   int res;
   res = fact(5);
   printf("%d", res);
   return 0;
}
```

Recursion

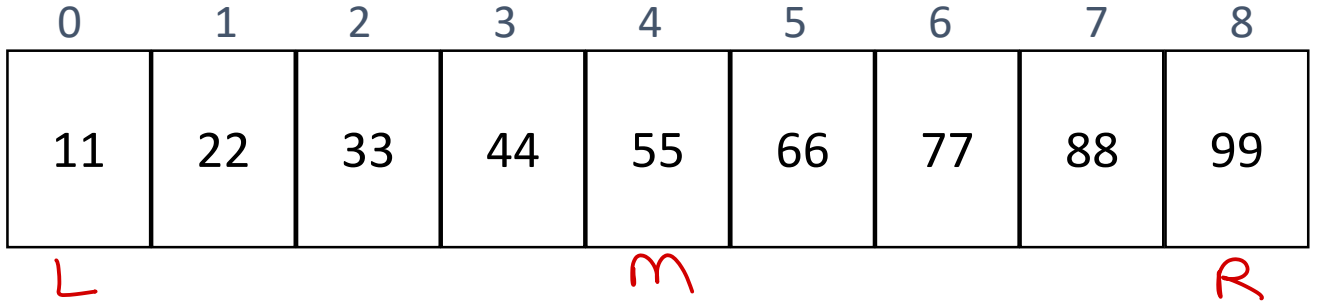
- × more space
- × more time
- ✓ Simplified coding

stack

- fact(0)
- fact(1)
- fact(2)
- fact(3)
- fact(4)
- fact(5)
- main()

Binary Search

Key = 44




```
int binSearch(arr, left, right, key) {
    if(left > right)
        return -1;
    mid = (left + right) / 2;
    if(key == arr[mid])
        return mid;
    if(key < arr[mid])
        idx = binSearch(arr, left, mid-1, key);
    else
        idx = binSearch(arr, midi, right, key);
    return idx;
}
```

Selection Sort

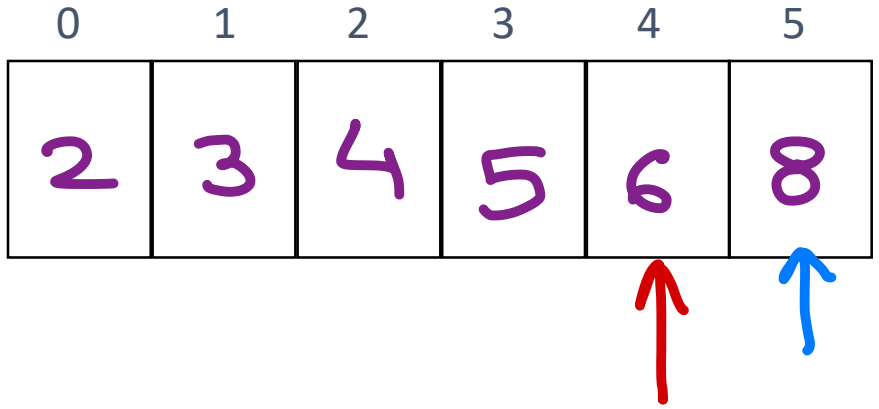
5 6 3 8 2 4

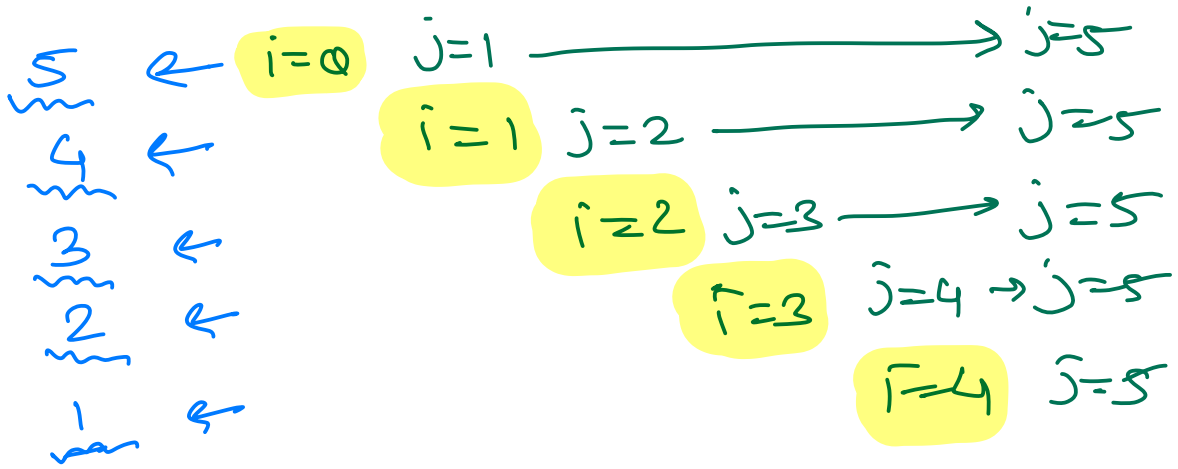
```
for(i=0; i<n-1; i++){
    for(j=i+1; j<n; j++){
        if(a[i] > a[j]){
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
```

3

3

3





Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>