Operating System

① what is an operating system?

↳ program which acts as an intermediary b/w the user and the computer hardware.

② Different operating systems.

↳ Multi-tasking

↳ Multi-Programming

↳ Real Time - operating System

↳ Batch operating

↳ Time - Sharing OS

↳ Distributed Operating System.

③ Basic functions of OS?

↳ controls and co-ordinates the use of hardware for various uses.

↳ acts as resource allocator & manager.

↳ controls the user programs to prevent errors and improper use of computer.

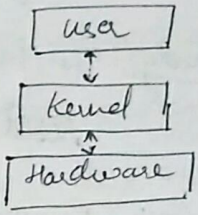
④ What is kernel?

↳ core & essential part of OS that provides basic services.

↳ is responsible for disk management, memory management, task management etc.

↳ first program to load after the bootloader.

↳ it is a bridge b/w the user and the hardware, it translates user instructions for the hardware and connects them.



⑤ Micro Kernel:

↳ runs services that are minimal for OS performance.

↳ all other operations are performed by processor.

Macro Kernel:

↳ combination of micro + monolithic kernel.

↳ all OS code is in single executable file

⑥ Deadlock

↳ condition where two processes are waiting for each other to complete so that they can start.

↳ this results both the processes to hang.

⑦ Process

↳ program in execution.

↳ Two types – OS processes
– user processes

⑧ States of a process

↳ New
↳ Running
↳ Wait
↳ Ready
↳ Terminate

① New: Program going to be picked up by the OS into the main memory

② Ready: The processes which are ready for the execution and reside in main memory.

③ Waiting/Running: One of the processes from the Ready state will be chosen by the OS depending on the scheduling algorithm.
The no. of running process will always be one at a time 'cause only one process will be picked up by the CPU for execution.

④ Wait: When a process waits for a certain resource to be assigned or input from the user, then the OS moves this process to the block/wait state and assigns the CPU to other processes.

(5) Terminate:

↳ when a process finishes its execution, it comes to the terminate state.

(9) Starvation

↳ it a resource management problem where a process does not get the resources it needs for a long time because the resources are being allocated to other processes.

Aging

↳ technique to avoid starvation in a scheduling algorithm.

(10) Semaphore

↳ variable whose states reports common resource.

↳ Two types – binary and counting semaphores.

Binary semaphores

↳ values restricted to 0/1.

↳ wait when 0

↳ signal when 1

Advantages

↳ allow only one process to be in the critical section.

↳ follow mutual exclusion.

↳ no resource wastage

Disadvantage:

↳ complicated.

↳ impractical for large scale use.

Counting Semaphores

↳ non negative integer values.

↳ coordinate the resource access.

↳ semaphore count = no. of available resources.

↳ incremented when resource added else subtracted.

Data Structures :

① Search for a target key in linked list?
↳ apply sequential search.
↳ each node is traversed and compared with a target key.
↳ traversal is continued until the key is found / last node is reached.

② Recursive algorithm
↳ a problem is divided into small sub problems.
↳ the O/p of one recursion after processing one sub-problem becomes the I/p to the next recursive process.

③ Fibonacci search
↳ algo. applied to a sorted array
↳ uses a divide-and-conquer approach to reduce time.

④ Huffman's algorithm
↳ creating extended binary trees that have minimum weighted path lengths.
↳ makes use of table that contains the frequency of occurence for each data element.

⑤ Doubly linked lists :
↳ traversal can be done in both directions
↳ each node has two links.

⑥ AVL Tree
↳ binary search tree in a state of partially balanced.
↳ balance is diff b/w the heights of the subtrees from the root.

29/04/2021

• Towers of Hanoi

↳ minimum no. of steps required with 'n' disks = $2^n - 1$

↳ Three towers - source, dest and aux.

↳ **Rules:**

① Only one disk can be moved among the towers at any given time.

② Only the "top disk" can be removed.

③ No large disk can sit over a small disk.

↳ **Steps:**

① Move $n-1$ disks from source to aux.

② Move n^{th} disk from source to dest.

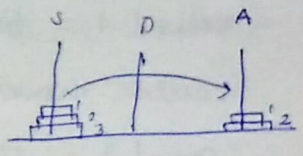
③ Move $n-1$ disks from aux to dest.

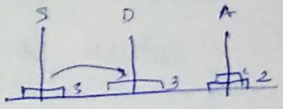
when $n = 3$,

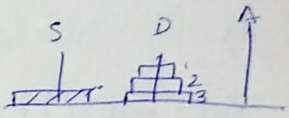
① Move 2 disks from source to aux.

② Move 3rd disk from source to dest.

③ Move 2 disks from aux to dest.







↳ **Algorithm:**

```
START
Procedure Hanoi (disk, source, dest, aux)
    If disk == 1, then
        move disk from source to dest.
    Else
        Hanoi(disk - 1, source, aux, dest)    // Step 1
        move disk from source to dest         // Step 2
        Hanoi(disk - 1, aux, dest, source)    // Step 3
    End If
End Procedure
STOP
```

• Code Walkthrough

```
#include<bits/stdc++.h>
using namespace std;

class Solution{
  public:
      long long toh(int N, int from, int to, int aux){
      int count=0;
Condition: if(N==1)
      {
        cout<<"move disk1 from rod"<<from<<"to rod"<<to
            <<endl;
        count++;
        return count;
      }
                          source      destination
  count+=toh(N-1, from, aux, to);
  cout<<"move disk"<<N<<"from rod "<<from<<"to rod "<<to<<endl;
  count+=toh(N-1, aux, to, from);
  return count+1;
  }
};

int main(){
    int T;
    cin>>T;
    while(T--){
     int N;
      cin>>N;
      Solution ob;
      cout<<ob.toh(N,1,3,2)<<endl;
}
```

① Sort an array of 0s, 1's and 2s.

Given an array A[] consisting of 0s, 1s and 2s. The Task is to write a function that sorts the given array.
The function should put all 0s first, then all 1s and all 2s in last.

Example: Input = {0, 1, 2, 0, 1, 2}.
Output = {0, 0, 1, 1, 2, 2}

Input = {0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1}.
Output = {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2}

Approach 1:

Count the number of 0s, 1s and 2s in the given array.
Store all 0s in the beginning followed by 1s and then 2s.

Algorithm

① Keep 3 counters, c_0 for 0, c1 for 1 and c2 for 2s.
② Keep increasing the count of these variables c_0 when 0 is found, c1 when 1 is found and c2 when 2 is found.
③ Replace first c_0 elements with 0, c1 elements with 1 and c2 elements with 2's. Print the array.

Time Complexity: $O(n)$

↳ we are only traversing the array once.

Space Complexity: $O(1)$

↳ no extra space is required.

Approach 2:

The algorithm is also known as Dutch National Flag or three-way partitioning.

Steps:

① Declare three variables low=0, high = n-1, mid=0, where n= length of the array.

② Run a loop until mid<= high, three cases on the basis of value of arr[mid].

① If arr[mid] == 0
swap(arr[low], arr[mid])
mid++;
low++;
break;

② If arr[mid] == 1
// keep at it is
mid++;
break;

③ If arr[mid] == 2
swap(arr[mid], arr[high])
high--;
break;

Time Complexity: $O(n)$

Space Complexity: $O(1)$

no extra space is required.