Lab Assignment - 1

Akhil P Dominic
MT23013

Q1.

2. Give the sequence of moves required to move 8 disks from source to destination, using the above algorithm. Explain how the code is working with 8 disks.

In a traditional tower of Hanoi setup, we would have 3 pegs. To solve that, we would first move the n-1 disks from the top of the source peg to the intermediate peg. Then, we would be moving the last disk in the source (which would be the largest) to the destination peg. After that, we would move the n-1 disks in the intermediate peg to the destination peg, thereby solving the tower of Hanoi problem.
If we have four pegs, then we would be having two intermediate pegs, which would reduce the time complexity required to solve the problem. So we will be moving the top n-k disks from the top of the source peg and move those disks to any of the intermediate pegs. Consider the value of k as k=2. After moving the n-2 pegs from the source to intermediate peg T2, we could move the second last peg to the other intermediate peg T3. Now we are left with the last peg in the source, which could be moved to destination peg T4.
Now, the disk at T3 could be moved to T4. At this stage, we have the n-2 disks at T2 and the 2 disks at the destination. Using recursion, we could solve the problem of the remaining n-2 pegs as well. Finally, we could get those n-2 disks on the last peg as well using recursion.

Suppose we are having 8 disks.
The sequence of steps are:-
T1 -> T3
T1 -> T4
T3 -> T4
T1 -> T2
T1 -> T3
T2 -> T3
T4 -> T1
T4 -> T3
T1 -> T3
T1 -> T4
T1 -> T2
T4 -> T2
T3 -> T2
T3 -> T4
T2 -> T4
T3 -> T1

T3 -> T2
T1 -> T2
T4 -> T3
T4 -> T2
T3 -> T2
T1 -> T3
T1 -> T4
T3 -> T4
T2 -> T3
T2 -> T1
T3 -> T1
T2 -> T4
T2 -> T3
T4 -> T3
T1 -> T2
T1 -> T3
T2 -> T3
T2 -> T1
T2 -> T4
T1 -> T4
T3 -> T4
T3 -> T1
T4 -> T1
T3 -> T2
T3 -> T4
T2 -> T4
T1 -> T3
T1 -> T4
T3 -> T4

This is the function of the tower of Hanoi with 4 pegs.
The total number of steps taken was 45.

```
void towerofhanoi4(int n,int k,int source_peg,int inter_peg1,int inter_peg2,int
destination_peg)
{

   Here we are checking if we have reached the last disk on the peg
   If yes, we would move that to the destination peg
   if(n==1)
   {

      cout<<"T"<<source_peg<<" -> T"<<destination_peg<<endl;
      Here we are increasing the step counter increasing
```

count_val++;
}

If the above condition is not true

else if(n>1)
{

Here we are recursively calling the towerofhanoi4 function
We would move the n-k disks from the source peg to one of the intermediate pegs.
We would be taking n=8 and k=2
In the case of 8 pegs, we would be moving the top 6 disks from the source peg to the intermediate peg 1.

towerofhanoi4(n-k,k,source_peg,inter_peg2,destination_peg,inter_peg1);

Here, we are moving the second last element in the source peg to the other intermediate peg 2.

cout<<"T"<<source_peg<<" -> T"<<inter_peg2<<endl;

Now moving the last element in the source to the destination peg.

cout<<"T"<<source_peg<<" -> T"<<destination_peg<<endl;

Finally moving the second last disk in the intermediate peg T2 to the destination peg.

cout<<"T"<<inter_peg2<<" -> T"<<destination_peg<<endl;

Here we are increasing the step counter
count_val+=3;

Now we are recursively calling the function to move the n-k(6) disks in the intermediate peg 1 to the destination peg.

towerofhanoi4(n-k,k,inter_peg1,inter_peg2,source_peg,destination_peg);
}
}

3. Compare the answer you have received in Q1.2 with the answer you will receive with the traditional Tower of Hanoi Setup (1 source pole, 1 temporary pole, 1 destination pole)I.e. compute and compare their time complexities.

In the traditional Tower of Hanoi setup, if we use the recursive tree method, initially the T(n) would be split into 2T(n-1)+1. Here, we initially had n disks and moved the n-1 top disks to the intermediate pole. After that, we move the last disk to the destination pole and the n-1 disks from the intermediate pole to the destination pole. So T(n) became 2T(n-1) problems along with the 1 for moving the largest disk from the source to the destination pole.

So,
$T(n)=2T(n-1)+1$

Now, the problem with n-1 disks would need solving the problem for 2T(n-2) and so on
$T(n-1)=2T(n-2)+1$

$T(n-2)=2T(n-3)+1$

On substituting, T(n-2) in T(n-1),
$T(n-1)=2(2T(n-3)+1)+1$
$T(n)=2(2(2T(n-3)+1)T+1)$

Similarly,
$T(n-k)=2T(n-k+1)+1$

So $T(n)=2^kT(n-k)+1$

WIth n-k=1,
k=n-1

So, time complexity would be
$T(n)= 2^k T(n-k) + 2^k-1+.....+2+1$

On substituting k=n-1

Time complexity would be of the order of $O(2^n)$

On the other hand, when we are using the tower of Hanoi with 4 pegs, we would be splitting the problem with complexity T(n) into 2T(n-x)+1 problems. We took x=2 for the solution.

So, the time complexity would be
$T(n)=2T(n-2)+1$
$T(n-2)=2T(n-4)+1$

$T(n)=2(2T(n-4)+1)+1$

$T(n-4)=2T(n-6)+1$

Similarly,

$T(n-2k)=2T(n-2(k-1))+1$

$n-2k=1$
$k=(n-1)/2$

So $T(n)=2^k T(n-k) + 2^{(k-1)}.....+2+1$

So the time complexity of the tower of Hanoi problem with 4 pegs is of the order $O(2^{(n/2)})$.

In the traditional tower of Hanoi, in order to move 8 disks, it took 255 steps. But in the case of the tower of Hanoi with 4 disks, it took only 45 steps to move 8 disks from source to destination. Thus, the tower of Hanoi with 4 pegs has a reduced time complexity.

Credits: The Four-Peg Tower of Hanoi Puzzle by I-Ping Chu Richard, Johnsonbaugh, ACM

Q2

2. Explain the iterative code by an example.

In iterative merge sort, we are splitting the given array into smaller and smaller parts and finally merging them in an iterative manner.

Suppose we have the following array:

8 7 6 5 4 3 2 1

In the iterative method, we would be first splitting the elements into groups of 2 and comparing the various groups and merging them.

So first we split the array into groups of 2:

We have groups [8,7], [6 5], [4,3], [2,1]
We sort and merge the arrays : [7,8], [5,6], [3,4], [1,2]

Now we have groups of 4,
[5,6,7,8], [1,2,3,4]

Then finally compare and merge into group of 8,
[1,2,3,4,5,6,7,8]

This is the function for iterative merge sort

```
void mergesortIterative(int arr[],int arr_length)
{
```

We declare the necessary variables

```
int low,high,mid,i,j;
```

Now we have a four loop that has initial value 2 and it increases the size of the each group to merge

```
for(i=2;i<=arr_length;i=i*2)
{
```

Looping through the elements in each group

```
for(j=0;j+i-1<arr_length;j=j+i)
 {
```

Setting low as the first element in the group and high as the last

```
low=j;
 high=j+i-1;

 mid=(low+high)/2;
```

Finding the mid element and merging

```
merge(arr,low,mid,high);
    }
  }
```

If at any time the array cannot be split into half, or we can have only maximum two groups, we merge them

```
  if(i/2<arr_length)
  merge(arr,0,i/2-1,arr_length-1);

}
```

Q3

2. Explain the above implementation with an example.

Here, we have 4 functions:

The **insertPQ(P arr[], int val, int priority)** function, we would be inserting the val and priority of the element to the array. On insertion, we would be placing the element based on its priority, so that the element with the highest priority could be easily obtained in O(1) complexity.

The **findPQ(P arr[],int val)** function, we would be finding the element with value val in the array and also returning its priority. We use Linear search for that.

The **deletePQ(P arr[], int val)** function, we would be deleting the element from the queue with the value val.

The **popPQ(P arr[])** would delete the element with the highest priority from the queue

Example:

Consider we want to input the following values to the priority queue:
14,3
11,6
34,1

We would use the insertPQ() function to insert the values.

We could use the findPQ() function to find the values and return their priority as well.

We can use deletePQ() to delete 11 from the priority queue.

We may then use popPQ() to delete the highest priority element from the queue.

Example output taken from my implementation :

>>Which operation do you want to perform: 1:Insert, 2:Find, 3:Delete, 4:Pop : 1
>>Please enter the value,priority to be inserted:14,3
>>Do you want to perform more operations? Enter 'Y' or 'N':

Y
>>Which operation do you want to perform: 1:Insert, 2:Find, 3:Delete, 4:Pop : 1
>>Please enter the value, priority to be inserted:11,6
>>Do you want to perform more operations? Enter 'Y' or 'N':
Y
>>Which operation do you want to perform: 1:Insert, 2:Find, 3:Delete, 4:Pop : 1
>>Please enter the value, priority to be inserted:34,1
>>Do you want to perform more operations? Enter 'Y' or 'N':
Y
>>Which operation do you want to perform: 1:Insert, 2:Find, 3:Delete, 4:Pop: 2
>>Please enter the value to be searched: 11
Value = 11 found with priority 6
>>Do you want to perform more operations? Enter 'Y' or 'N':
Y
>>Which operation do you want to perform: 1:Insert, 2:Find, 3:Delete, 4:Pop : 3
Enter the value to deleted: 11
>>Do you want to perform more operations? Enter 'Y' or 'N':

N
>>Program ended. 2 element remaining in the queue.