Analyse of the programs

2.

T(N)=T(N-1)+1 –(1)

T(N-1)=T(N-2)+1 –(2)

T(N-2)=T(N-3)+1 –(3)

Substitute

T(N)=T(N-2)+1+1

T(N)=T(N-3)+1+1+1

T(N)=T(N-K)+K

If k=n

T(N)=O+N;

O(N) time complexity

3.

T(N)=T(N-1)+T(N-2)+C

Considering T(N-1)~T(N-2)

T(N)=2\*T(N-2)+C

T(N)=2\*(2\*T(N-4)+C)+C

T(N)=4T(N-4)+3C

T(N)=2^K\*T(N-2^K)+(2^K-1)\*C

N-2K=0

K=N/2

T(N)=2^(N/2)\*T(N-2^(K/2))+(2^(N/2)-1)\*C

O(N) time complexity

4.

Recursive-

T(N,K)=T(N-1,K)+T(N-1,K-1)+0(1)

If K=1

T(N,N)=T(N,0)=O(1)

N is decreased by one every step if we ignore that there is parameter k basically number of cells doubles every steps this happens n times enter n=1

Now C(1,x) return 1 so (n,x) will call 2^n times.

T(N,X)=O(2^N)

Now we remember the k and worst case for k may be n/2 so you need at least n/2 steps until k reduces 1 or n reaches n/2 so number of calls doubles at least 2^(n/2) times.

C(N,N/2)=2^N

Interactive-

There are 2 for loop n and m so it will take O(N\*K) times and N\*K space.

5.

Selection sort requires the tested for loops to complete itself, one loop is in the function Section Sort, and inside first loop we are taking a call to another function index of minimum, which has second(inner) for loop.

Hence for a given input n.

Worst Case – O(n^2)

Best Case – O(n^2)

Average Case – O(n^2)

6.

Iterative-

Log(N) time for iteration

Power function take log(N) time

In pow(a,b), b=log n

So log(log n) for pow function

So total complexity= log n log(log n)

Recursive-

O(log n log(log n) )

7.

Time taken T(N)=T(N-1)+1+T(N-1)

=2\*T(N-1)+1

T(N-1)=2\*T(N-2)+1

T(N)=2\*(2\*T(N-2)+1)+1

T(N)=2^2\*T(N-2)+2+1

Put N=N-2

T(N-2)=2\*T(N-3)+1

T(N)=2^2(2\*T(N-3)+1)+2+1

T(N)=2^3T(N-3)+2^3+2^2+…

T(N)=(2^I)\*T(N-I)+(2^(I-1))+(2^(I-2))+……

N-I=1;

T(N-I)=1

Now , T (N)=2^I+2^(I-1)+2^(I-2)+…..

Using GP

S=(a(1-(r^n)))/(1-r)

=i(1-(2^(i+1)))/(1-2)

T(N)=2^(I+1)-1

T(N)=2^(N-1+1)-1

T(N)=2^N-1

10.

When a is divisible by b, remainder is always less than or equal to a/2 because if remainder is more than a/2, and since divisor is always greater than remainder then divisor is also more than a/2 and so sum of divisor and remainder becomes more than a, which cannot be possible.

Now when we find gcd(a,b) suppose a>b if not swap a and b in first step a is dividend and b is divisor, we find some remainder r1, then in second step r1 become divisor and b become dividend. Now again we divide b by r1 and get some remainder r2 but due to above property r2<=b/2.

So in two steps, remainder is at most b/2 we terminate the process once we remainder to 0. In the worst case, every 2 step reduces remainder to b/2 and then we need log2b such steps or total 2log2b steps.

So gcd(a,b) requires at most 2log2b recursive calls where b is min(a,b).

11.

This is looping O to n and a, b are taking case of memorization so it will take O(N) time.

14.

To do average case analyses, we need to consider all possible permutation of array and calculate the taken by every permutation which does not loop easy

So we can get an idea of average case by considering the case when positions puts O(N/9) elements in one set and O(9N/10) elements in set. Following is recursive for this case.

T(N)=T(N/9)+T(9N/10)+0(N)

Solution if above recursive is O(n logn)