DAA assignment #2

Ayank Jain
196101 BTech CSE VC

Ans 1> j i
1 0
2 0+1
3 0+1+2
⋮
K+1 0+1+2+...+K

let the loop runs for K times and base condition is/termination condition is $i=n$,

$\Rightarrow 0+1+2+\dots+K=n$

$\frac{K(K+1)}{2}=n \Rightarrow K \approx \sqrt{n}$

so time complexity is $O(\sqrt{n})$

Ans 2> Recurrence relation for fibonacci series is

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

for finding upper bound let us assume that

$T(n-2) \approx T(n-1)$, as $O(T(n-2))=T(n-1)$

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

using master's theorem, for $T(n)=aT(n-b)+O(g(n))$

here $a,b>0$ and $f(n)=O(g(n)^1)$

and $a>1$, so, time complexity is $O(2^{n/1}\times 1)=O(2^n)$

There will be $n-1 \approx n$ function calls which are stored in stack, so space complexity is $O(n)$.

Ans 3> for n(logn) complexity.

```
int fun(int arr[], int n)
{ if(n==1)
    return 0;
  int i, max=INT_MIN, mxind=-1;
  for(i=0;i<n;i++)
      if(arr[i]>max)
        { max=arr[i];
          mxind=i;
        }
  fun(n/2, arr);
  fun(n/2, arr);
}
```

for n3,

```
for (int i=0; i<n; i++)
   for (j=0; j<n; j++)
   { arr[i][j]=0;
     for (k=0; k<n; k++)
       arr[i][j] += arr1[i][k] * arr2[k][j];
   }
```

for log(logn)

```
p=0;
for (int i=0; i<n; i=i*2)
{ p++;
}
for (int j=0; j<p; j=j*2)
{
   cout<<p;
}
```

Ans 4) Since $T(\frac{n}{2}) \; \frac{n}{2} > \frac{n}{4}$, we can write $T(\frac{n}{4}) = T(\frac{n}{2})$ for finding upper bound of relation.

So, $T(n) = 2T(\frac{n}{2}) + cn^2$

Using master's theorem $\log_2 2 = 1$

$n^1 < n^2$, so time complexity will be $O(n^2)$

Ans 5)

i	j	
1	1+2+.. n	n times
2	1+3+.. n	$\frac{n}{2}$ times
3	1+4+.. n	$\frac{n}{3}$ times
⋮		
K		$\frac{n}{K}$ times

$\frac{n}{K} = 1 \Rightarrow n = K$ for base case

So, time will be $= n + \frac{n}{2} + \frac{n}{3} + .. + \frac{n}{n}$

$= n + (\frac{n}{2}) + \frac{n}{3} + .. + 1$

$= n\left(1 + \frac{1}{2} + \frac{1}{3} + .. + \frac{1}{n}\right)$

$= n \log n$

Ans 6→ i
2
2^k
2^{2k}
2^{3k}
$\vdots$
2^{pk}

base case, $2^{pk} = n$

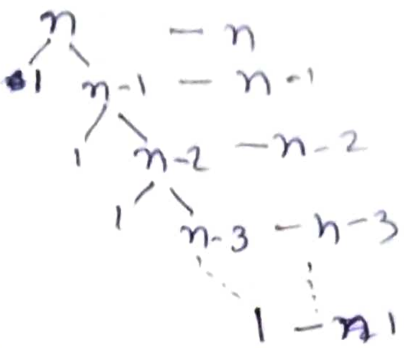
$pk = \log_2 n$

$p = \frac{\log_2 n}{k}$, where k is a constant

So timecomplexity $= O(\log n) = O(\lg n)$

Ans 7→ $T(n) = T(n-1) + n$

recurrence tree,



Time complexity $= n + (n-1) + (n-2) \ldots + 3 + 2 + 1$

$= \frac{n(n+1)}{2} \approx n^2$

so time complexity $= O(n^2)$

Ans 8→ a) $100, \log(\log n), \sqrt{n}, \log n, \log^2 n, n, \log(n!), n\log(n), n^2, 2^n, 4^n, n!, 2^{2^n}$.

b) $1, \sqrt{\log n}, \log(\log n), \log(n), \log(2n), 2\log(n), 3n, 2n, 4n, \log n!, n\log n, n^2, 2(2^n), n!$

c) $96, \log_8 n, \log_2 n, 5n, n\log_6(n), \log(n!), n\log_2(n), 8n^2, 7n^3, 8^{2n}, n!$