DAA

VINCE JOSEPH
Roll: 59
S3 MCA

05-10-20

1.) $f(n) = 5n^2 + 3n + 2^n$

$= 2^n + 5n^2 + 3n$

$f(n) \leq c \cdot g(n) \Rightarrow f(n) = O(g(n))$

$2^n + 5n^2 + 3n \leq 2^n + \underbrace{5n^2 + n^2}_{6n^2} \Rightarrow n > 3$

~~$2n^2 + 6n^2 \leq 2n$~~

$2^n + 6n^2 \leq \underbrace{2^n + 2^n}_{2 \cdot 2^n} = 2 \cdot 2^n$

$\Rightarrow c = 2, n_0 = 3, O(2^n)$

2.) ~~$2^{1024} < 100n < \log n$~~

$2^{1024} < \log n < 100n < n^2 < 2^n$

3.)

control abstraction for divide and conquer

Involves following steps

→ Divide
→ conquer
→ combine.

Let the problem be P

```
Algorithm    Divide conquer
{
    if small(P) then return Solution(P)
                    // the lowest sub problem
    else
    {
        Divide P into smaller subtasks like
            P1, P2 .... Pk , k > 1

        Apply divide conquer of each Pi above

    return Combine solutions of subtasks DivideConquer(P1),
            DivideConquer(P2) .... DivideConquer(Pk).
    }
}
```

VINCE

4.) Quick sort works in worst case complexity if array elements are already sorted (either ways) and we choose 1st or last element as pivot.

Or more specifically if we pick the ontseme element of that array as pivot.

Best case occures when the array is divided into two by the middle element / specifically the ~~Algo~~ partition process always picks the middle element as pivot $T(n) = 2T(n/2) + \Theta(n)$

6.)

i) $T(n) = T(3/4) + 1$

by master's theorem 1st find,

$\frac{n^{\log_b a}}{\Downarrow}$ here $a = 1$, $b = \frac{4}{3}$

$= n^{\log_{\frac{4}{3}} 1} = n^0 = 1 = f(n)$ | $\log 1 = 0$

so we have to use the 2nd condition of master's theorem ie if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log n)$

ie $T(n) = \Theta(\log n)$

VINCE

(iii) $T(n) = 25\,T(n/5) + n^4$

here $a = 25$, $b = 5$

$$n^{\log_b a} = n^{\log_5 25} \Rightarrow n^{\log_5 5^2}$$

$$\Rightarrow n^{2\frac{\log 5}{\log 5}}$$

$$= n^2$$

$f(n) = n^4$, so $n^{2+\epsilon}$ so use 3rd condition of master's theorem.

ie if $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ $T(n) = \Theta(f(n))$

if $a.f(n/b) \le c.f(n)$

$c < 1$

$$a.f(n/b) = 25.\left(\frac{n}{5}\right)^4 \le c.f(n)$$

$$\Rightarrow \frac{25}{5^4}.n^4$$, we can write $25.\left(\frac{n}{5}\right)^4 \le \frac{25}{625}.n^4$

$$\le \frac{1}{25}.n^4$$

so $T(n) = \Theta(n^4)$

VINCE

8.)

```
Algorithm maxmin (i, j, max, min)
// a[1:n] is an array, i, j are int, 1≤i≤j≤n
{
if (i == j) then max = min = a[i]  // small (P)
else if (i == j-1) then  // another small(P)
  {
      if( a[i] <a[j])then
      {
          max = a[j];
          min = a[i];
      }
      else
        {
          max = a[i];
          min = a[j];
        }
}
else
{
    // P is not small so divide P
    mid = (i+j)/2;
    maxmin (i, mid, max, min);
    maxmin (mid+1, j, max1, min1);
    // combine
    if( max < max1 ) then  max = max1;
    if (min <> min1) then  min = min1;
  }
}
```

vinee

The idea of maxmin algorithm is to find the lowest & largest numbers in a given array.

for that we use recursive algorithm.

we use an array a[1:n], the given array.

i, j are used to pass lowest & highest index values

max, min variables used to return the minimum and maximum from an array.

The algorithm 1st checks for the smallest possible problem

ie when array has only 1 element / two element

if so the max and min are set to a[i] for 1 element array.

for 2 element array, a comparison is needed.

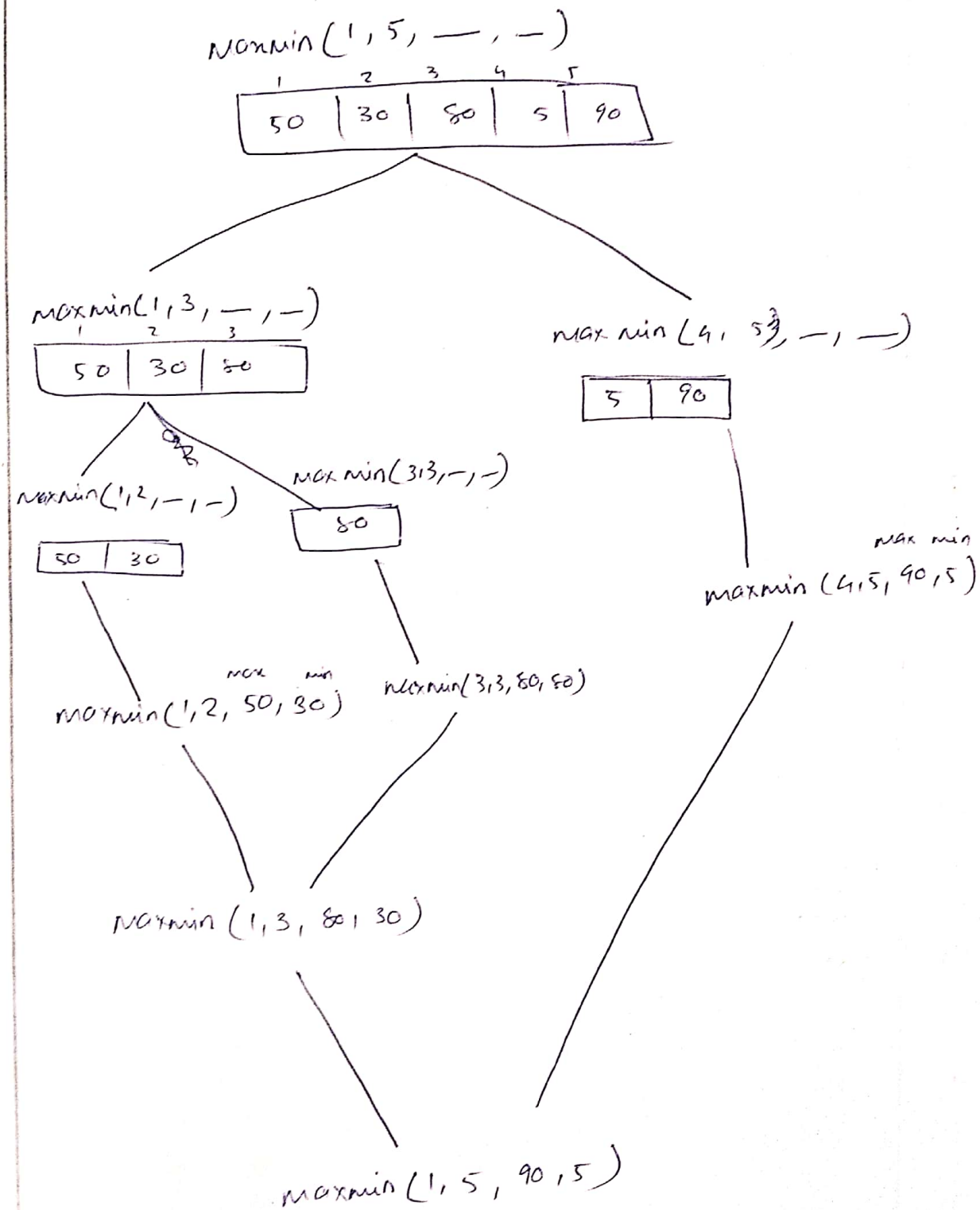
else it indicates that the problem is not small so we call the recursive function for both sides of the array spliting by its mid index.

The last we will combine the max and min values by checking appropriate conditions.

eg: given array

	1	2	3	4	5
a:	50	30	80	5	90

The algorithm will split the array as follows:



so max = 90
min = 5