PRAKSHEP GUSAIN

F : 19

Date

2016907 · Tutorial - 1

1) Asymptotic Notations are mathematical notations used to describe the running time of an algo. when input tends towards a particular or limiting value. Big O, Big Ω, Big Θ are different types of asymptotic notation.

2) 2^0 $i = 1$

2^1 $i = 2$

2^2 $i = 4$

⋮ ⋮

2^k ~~times~~ (k times)

$$\Rightarrow 2^k = n$$

$$\log_2 2^k = \log_2 n$$

$$k \log_2 2 = \log_2 n$$

$$k = \log_2 n$$

$$\therefore \text{Time Complexity} = O(\log n)$$

3) $T(n) = [3T(n-1)$ if $n > 0$ otherwise $1]$

$T(0) = 1$ $\quad n = n-1$

$$\Rightarrow T(n-1) = 3T(n-1-1) = 3T(n-2)$$

$$\therefore T(n) = 3[3T(n-2)]$$

$$n = n-2 \Rightarrow T(n-2) = 3T(n-3)$$

$$\therefore T(n) = 3[3 \cdot 3T(n-3)]$$

$$\Rightarrow T(n) = 3^k T(n-k)$$

$$n - k = 0 \Rightarrow n = k$$

$$T(n) = 3^n T(0) = 3^n(1) = 3^n$$

$$\therefore T.C. = O(3^n)$$

Teacher's Signature

4) $T(n) = 2T(n-1) - 1 \qquad T(0) = 1$

$n = n-1 \Rightarrow T(n-1) = 2T(n-2) - 1$

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

$n = n-2 \Rightarrow T(n-2) = 2T(n-3) - 1$

$T(n) = 8T(n-3) - 4 - 2 - 1$

$n = n-3 \Rightarrow T(n-3) = 2T(n-4) - 1$

$T(n) = 16T(n-4) - 8 - 4 - 2 - 1$

$\vdots \qquad \vdots$

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - \dots - 2 - 1$

$n - k = 0 \Rightarrow n = k$

$\Rightarrow T(n) = 2^n T(0) - [2^{n-1} + 2^{n-2} + 2^{n-3} + \dots + 2 + 1]$

$= 2^n - [1 + 2 + 2^2 + \dots + 2^{n-1}]$

$= 2^n - [1 + (2 + 2^2 + 2^3 + \dots + 2^{n-1}]$ $\quad \cancel{= 2^n - \frac{(n-1)(n-2)}{2} - 2^n + 1}$

$= 2^n - \left[1 + \frac{2(2^{n-1} - 1)}{2 - 1}\right] + 1$

$= 2^n - [\cancel{1} + 2^n - \cancel{1}] + 1$

$= \cancel{2^n} - \cancel{2^n} + 1$

$\Rightarrow T.C. = O(1)$

5) $i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad \dots$

$s = 1 + 3 + 6 + 10 + 15 + \dots$

Sum of $S = 1 + 3 + 6 + + 10 + \dots + n$ — ①

Also $S = 1 + 3 + 6 + \dots + T_{n-1} + T_n$ — ②

$0 = 1 + 2 + 3 + 4 + \dots n - T_n$

$T_k = 1 + 2 + 3 + 4 + \dots + k$

Date Date Page No.

$T_K = \frac{1}{2} K (K+1)$

for K iterations

$1 + 2 + 3 + \_\_ + K \quad <= n$

$\frac{K(K+1)}{2} <= n$

$\frac{K^2+K}{2} <= n$

$O(K^2) <= n$

$\cancel{K <=} \quad K = O(\sqrt{n})$

$\therefore \boxed{T(n) = O(\sqrt{n})}$

6) $i^2 = n \Rightarrow i = \sqrt{n}$

$i = 1, 2, 3, \_\_ \sqrt{n}$

$\sum_{i=1} 1 + 2 + 3 + \_\_ + \sqrt{n}$

$T(n) = \frac{\sqrt{n} + (\sqrt{n} + 1)}{2}$

$T(n) = (n + \sqrt{n})/2$

$T(n) = O(n)$

7) for $K = K^2$, $K = 1, 2, 4, 8, \_\_ n$

$a = 1, \quad r = 2$

$a\frac{(K^n - 1)}{K - 1} = \frac{1(2^K - 1)}{1}$

$n = 2^K - 1 \Rightarrow K = \log_2 ($

~~Tx~~	i	j	K
	1	log(n)	log(n) * log(n)
	2	log(n)	log(n) * log(n)
	⋮	⋮	⋮
	n	log(n)	log(n) * log(n)

$T.C. = O[n * \log n * \log n]$

$= O(n \log^2(n))$

8) for (i=1 to n)

We get $j = n$ times every turn

$\therefore i * j = n^2$

Now, $T(n) = n^2 + T(n-3)$

$T(n-3) = (n^2 3)^2 + T(n-6)$

$T(n-6) = (n^3 6)^2 + T(n-9)$

and $T(1) = 1$

Now substitute each value in $T(n)$

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \cdots + 1$

Let $k^n - 3k = 1$

$k = (n-1)/3$ total time $= k+1$

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \_\_ + 1$

$T(n) \simeq n^2$

$T(n) \simeq (k-1)/3 + n^2$

So $T(n) = O(n^3)$

9) $i=1$ $\quad j = 1+2+ \ldots \ldots (n \geq j+i)$

$i=2$ $\quad j = 1+3+5 \ldots \ldots (n \geq j+i)$

$i=3$ $\quad j = 1+4+7 \ldots \ldots (n \geq j+i)$

nth term of AP is

$T(n) = a + d * m$

$T(n) = 1 + d * m$

$(m-1) d = n$

for $i=1$ $\quad (n-1)/1$ times

$i=2$ $\quad (n-1)/2$ times

$i=n-1$

We get,

$T(n) = i_1 j_1 + i_2 j_2 + \ldots + i_{n-1} j_{n-1}$

$= \frac{n-1}{2} + \frac{n-2}{2} + \ldots + 1$

$= n [1 + n/2 + n/3 + \ldots n/n-1$

$= n [1 + 1/2 + 1/3 + \ldots + 1/n-1] -$

$= n \times \log n - n + 2$

Since $\int 1/x = \log x$

$T(n) = O(n \log n)$

10) Assume $k >= 1$ & $c > 1$ are const.

$\Rightarrow n^k = O(c^n)$

$n^k = a(c^n) \quad \forall \quad n \geq n_0$ & a

for $n_0 = 1$, $c = 2$

$1^k < a^2$

$\Rightarrow n_0 = 1$ & $c = 2$

Tutorial - 2

1)

$j = 1 \quad i = 1$

$j = 2 \quad i = 1 + 2$

$j = 3 \quad i = 1 + 2 + 3$] m - level

for (i)

$\because 1 + 2 + 3 + \ldots + < n$

$\because 1 + 2 + 3 + m < n$

$\because \dfrac{m(m+1)}{2} < n$

$$m \approx \sqrt{n}$$

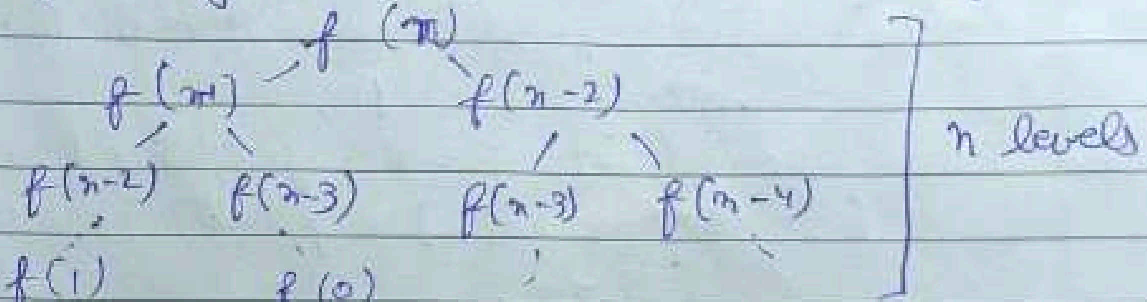
$\sum_{i=1}^{m} 1 \Rightarrow 1 + 1 + \text{---} + \sqrt{n}$ times

$$\boxed{T(n) = \sqrt{n}}$$

2) For fibnacci series,

$f(n) = f(n-1) + f(n-2)$ $\qquad f(0) = 0$

By forming a tree, $\qquad f(1) = 1$



$\because$ for every fun call, we get 2 fun. calls, $\therefore$ for n levels, we have

$2 \times 2 \times \text{--}$ n times

$\therefore T(n) = 2^n$

<u>MAXIMUM SPACE</u> - considering Recursion stack: no. of max. calls $= n$

For each call, S. C. $= O(1)$

$\therefore \boxed{T(n) = O(n)}$

without Recursive stack,

$\boxed{T(n) = O(1)}$

3) (i) $n \log n \Rightarrow$ Quick sort

```
void quicksort (int arr [], int l, int h)
{   if (l < h)
   { int pi = partition (arr, l, h);
     quicksort (arr, l, pi -1);
     quicksort (arr, pi+1, h);
} }
int partition (int arr [], int l, int h)
{   int pivot = arr [h];
    int i = (l-1);
  for (int j = l ; j <= h - 1; j ++)
  {   if (arr [i] < pivot)
       { i ++;
         swap (& arr [i], & arr [j]);
  }    }
     swap (& arr [i+1], & [h]);
     return (i+1);
}
```

(ii) n^3 → Triple loop

```
for (i=0; i < n ; i++)
{ for ( j =0; j <n ; j ++)
{ for ( k=0 ; k < n ; k++)
  {   s = s + 1;
  }
}}
```

(iii) log (log n)

```
for (i=2 ; i < n ; i = i * i)
  { c++;
  }
```

4) $T(n) = T(n/4) + T(n/2) + cn^2$

n → 0

$T(n/4)$ $T(n/2)$ → 1

$T(n/3)$ $T(n/16)$ $T(n/4)$ $T(n/8)$ → 2

<u>At level</u> :

$0 \rightarrow cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{c5n^2}{16}$

$2 \rightarrow \frac{n^2}{8^2} + \frac{n^2}{16^2} + \frac{n^2}{4^2} + \frac{n^2}{8^2} = \left(\frac{5}{16}\right)^2 n^2 c$

⋮

max level $= \frac{n}{2^k} = 1$

$= k = \log_2 n$

Date

Page No.

$T(n) = c[n^2 + (\frac{5}{16})n^2 + (\frac{5}{16})^2 n^2 + \dots + (\frac{5}{16})^{\log n}]$

$T(n) = cn^2 \times 1 \times \left(\frac{1-(5/16)^{\log n}}{1-(5/16)}\right)$

$= cn^2 \times \frac{11}{5} \times \left[1-\left(\frac{5}{16}\right)^{\log n}\right]$

$T(n) = O(n^2 c)$

$O(cn^2)$

5) for

i	j
1	1
2	1+3+5
⋮	⋮
n	

$j = \frac{(n-1)}{i}$

$\sum_{i=1}^{n} \frac{n-1}{i} \Rightarrow T(n) = \frac{n-1}{1} + \frac{n-1}{2} + \frac{n-1}{3} + \dots$

$T(n) = n\left[1 + \frac{1}{2} + \frac{1}{3} + \dots \frac{1}{n}\right] - 1 \times \left[1 + \frac{1}{2} + \dots\right.$

$= n \log n - \log n$

$T(n) = O(n \log n)$

6) for i

2^1

2^k

2^{k^2}

⋮

2^{k^m}

where

$2^k <= n$

$k^m = \log_2 n$

$m = \log_k \log_2 n$

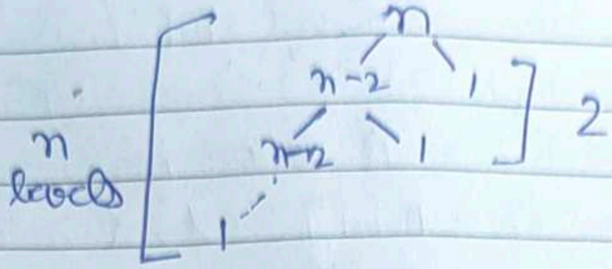
$1+1+1+1 \dots\ m$ times

$T(n) = O(\log_k \log n)$

Mohan

Teacher's Signature

7) $\because T(n) = T(n-1) + O(1)$



'n' work is done at each level

$T(n) = [T(n-1) + T(n-2) + \dots + T(2) + O(1)] \times n$
$= n \times n$
$\therefore T(n) = O(n^2)$

Lowest height $= 2$
Highest height $= n$
$\therefore$ Difference $= \underline{\underline{n-2}}$ $\quad n > 1$

Given algo gives linear result

8) (a) $100 < \log \log n < \log n < (\log n)^2 < \sqrt{n} < n < n \log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2^n}$

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

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

Date

Page No.

Tutorial 3

1)
```
int linear_search (int A[], int n, int t)
{ if (abs [A[0] - t) > abs (A[n-1] - t))
      for (i = n-1 to 0; i--)
          if (A[i] == t) { return i; }
  else for (i = 0 to n-1; i++)
          if (A[i] == t)
              return i;
}
```

2)
```
void insertion (int A[], int n)
{ for (i = 1 to n)
    { t = A[i];   j = i;
    while (j >= 0 && t < A[j])
      { A[j+1] = A[j];
             j --;
    } A[j+1] = t;
} }
```

Insertion sort is also called online sorting algo. because it will work if elements to sorted are provided one at a time with the understanding that algo must keep seq sorted as more elements are added

3)

	Best	Worst
Bubble Sort	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$

Hohan

Teacher's Signature

Page No.

Insertion Sort	$O(n)$	$O(n^2)$
Count Sort	$O(n)$	$O(n+k)$
Quick Sort	$O(n \log n)$	$O(n^2)$
Merge Sort	$O(n \log n)$	$O(n \log n)$
Heap Sort	$O(n \log n)$	$O(n \log n)$

4)

	Inplace	Stable	Online
Bubble	✓	✓	✗
Selection	✓	✗	✗
Insertion	✓	✓	✓
Count	✗	✓	✗
Quick	✓	✗	✗
Merge	✗	✓	✗
Heap	✓	✗	✗

5) <u>Recursive</u>:

```
int Binary (int arr [], int l, int r, int x)
{   if (r >= l)
    { int mid = l + (r-l)/2;
      if (arr [mid] == x)  return mid;
      else if (arr [mid] > x)
        return Binary (arr, l, m-1, x)
        else return Binary (arr, mid+1, r, x);
    }
    return (-1);
}
```

Teacher's Signature

Date

Page No.

Iterative -
```
int Binary (int arr[], int x)
{ int l=0, r = arr.length -1;
  while (l <= r)
  { int m = l + (r-1)/2;
    if (arr [m] == x) return m;
    if (arr [m] < x)   l = m+1;
    else   r = m-1;
  } return -1;
}
```

6)

$T(n)$
$\downarrow$
$T(n/2)$
$\downarrow$
$T(n/4)$
$\vdots$
$T(n/2^k)$

Recurrence Relation
$= T(n/2) + O(1)$

7)
```
int n;  int key;
int A[n];  int i=0, j = n-1;
  while (i < j)
  { if (A[i] + A[j]) == key)
      break;
    else if ((A [i] + A[j]) > key)
      j --;
    else  i ++;
  } cout << i << " " << j;
```

Time Complexity $= O(n \log n)$

Mohan

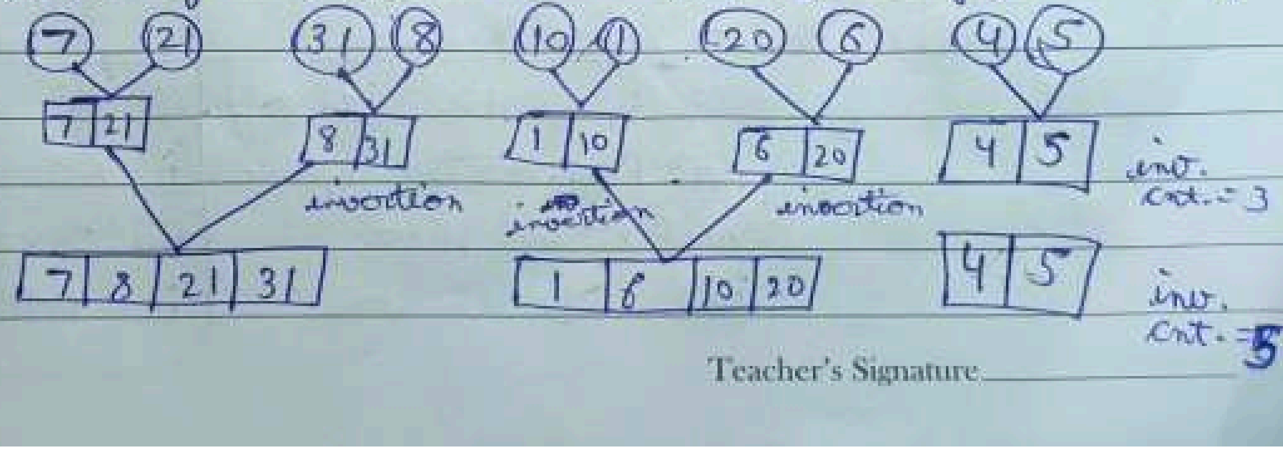
Teacher's Signature

8) Factors affecting sorting algo is good or not are:
(i) Run Time
(ii) Space
(iii) Stable
(iv) No. of Swaps
(v) will data fix in RAM

Does data fit in RAM?
- yes → Are swaps expensive
  - yes → Selection Sort
  - No → Is data sorted?
    - yes → Insertion
    - No → Can we use extra space?
      - yes → Does it need to be stable
        - yes → Merge
        - No → Quick
      - No → Quick
- No → Merge

9) ~~Insertion~~ Inversion indicates how far array is from being sorted. If array is sorted, inversion count is 0 but if it is reverse sorted, inversion count is maximum.

Cond. for inversion: $a[i] > a[j]$ and $i < j$



Date

1	6	7	8	10	20	21	31

4	5

inv. cnt = 0

1	4	5	6	7	8	10	20	21	31

inv count = 31

10) Best Case
T. C. = $O(n \log n)$
It occurs when partition process always picks mid. element as pivot.

Worst Case
T.C. = $O(n^2)$
when array is sorted in ascending / descen order.

11) Best Cases: Merge Sort = $2T(n/2)+n$
Quick Sort = $2T(n/2)+n$

Worst Cases: Merge Sort = $2T(n/2)+n$
Quick Sort = $T(n-1)+n$

Similarities - Both work on concept of divide and conquer algo.

Differnces -

Merge	Quick
(i) Array div. into 1/2	(ii) Div. in any ratio
(ii) Req. extra space	(ii) No extra space
(iii) External algo	(iii) Internal algo
(iv) Stable	(iv) Not Stable

13)
```
void Sort (int A [], int n)
{ int i, j;
   int f = 0;
  for (i = 0; i < n; i++)
 { for (j = 0; j < n-1; j++)
  { if (A [j] > A [j+1])
   { swap (A [j], A [j+1])
      f = 1;
   } if (f == 0)
       break;
 }
```

External Sorting -

Sorts massive amt. of data. Required when data does not fit inside RAM.

During sorting, chuncks of small data that can fit in main memory are read, sorted and written out to a temporary file.

Internal Sorting -

Used when entire collection of data is small enough to reside within RAM.

No need of external memory for program executions.

eg : Insertion sort, Quick sort

Date

Page No.

Tutorial - 4

1) $T(n) = 3T(n/2) + n^2$

$a = 3,\ b = 2,\ f(n) = n^2$

$c = \log_2 3 = 1.584$

$\Rightarrow n^c = n^{1.584} < f(n)$

$\therefore T(n) = \theta(n^2)$

2) $a = 4,\ b = 2,\ f(n) = n^2$

$c = \log_2 4 = 2$

$n^c = n^2 \quad f(n) = n^2$

$\Rightarrow T(n) = \theta(n^2 \log_2 n)$

3) $a = 1 \quad b = 2 \quad f(n) = 2^n$

$c = \log_2 0 = 0$

$n^c = n^0 = 1 < 2^n$

$\Rightarrow T(n) = \theta(2^n)$

4) $a = 2^n \quad b = 2,\ f(n) = n^2$

$c = \log_b a = \log_2 2^n$

$= n$

$n^c = n^n$

$\Rightarrow T(n) = \theta(n^2 \log_2 n)$

Date

6) $a = 2, b = 2, f(n) = n \log n$
$c = \log_2 2 = 1$
$n^c = n^1 = n < n \log n$
$\Rightarrow T(n) = \Theta(n \log n)$

7) $T(n) = 2T(n/2) + n/\log n$
$\Rightarrow c = \log_2 2 = 1$
$n^c = n^1 = n > \frac{n}{\log n}$

$\Rightarrow T(n) = \Theta(n)$

8) $T(n) = 2T(n/4) + n^{0.51}$
$\Rightarrow c = \log_4 2 = 0.5$
$n^c = n^{0.5} < n^{0.51}$
$\Rightarrow T(n) = \Theta(n^{0.51})$

9) $a = 0.5, b = 2$
$a \geq 1$ but here it is 0.5
so Master's Th. cannot be applied

10) $a = 16, b = 4, f(n) = n!$
$c = \log_4 16 = 2$
$n^c = n^2 < n!$
$\Rightarrow T(n) = \Theta(n!)$

11) $a = 4, b = 2, f(n) = \log n$
$c = \log_2 4 = 2$
$n^c = n^2 > \log n$
$\Rightarrow T(n) = \Theta(n^2)$

12) $a = \sqrt{n}, b = 2, f(n) = \log n$
$c = \log_2 \sqrt{n} = \frac{\log_2 n}{2}$

$n^c = n^{\frac{1}{2}\log_2 n} < f(n)$
$\Rightarrow T(n) = \Theta(\log n)$

13) $T(n) = 3T(n/2) + n$
$c = \log_2 3 = 1.5849$
$n^c = n^{1.5849} > n$
$\Rightarrow T(n) = \Theta(n^{1.5849})$

14) $a = 3, b = 3$
$c = \log_3 3 = 1$
$n^c = n^1 = n > \mathrm{sqrt}(n)$
$\Rightarrow T(n) = \Theta(n)$

15) $a = 4 \quad b = 2$
$c = \log_2 4 = 2$
$n^c = n^2 > n$
$\Rightarrow T(n) = \Theta(n^2)$

16) $a = 3 \quad b = 4$

$c = \log_4 3 = 0.792$

$n^c = n^{0.792} < n \log n$

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

17) $a = 3,\ b = 3,\ f(n) = n/2$

$c = \log_3 3 = 1$

$n^c = n > n/2$

$\Rightarrow T(n) = \Theta(n)$

18) $a = 6,\ b = 3$

$c = \log_b a = \log_3 6 = 1.6309$

$n^c = n^{1.6309} < n^2 \log n$

$\Rightarrow T(n) = \Theta(n^2 \log n)$

19) $a = 4,\ b = 2,\ f(n) = \frac{n}{\log n}$

$c = \log_2 4 = 2$

$n^c = n^2$

$\frac{n}{\log n} < n^2$

$\Rightarrow T(n) = \Theta(n^2)$

20) $a = 64,\ b = 8$

$c = \log_8 64 = \log_8 (8)^2 = 2$

$\Rightarrow n^c = n^2 < n^2 \log n$

$T(n) = \Theta(n^2 \log n)$

21) $a = 7,\ b = 3,\ f(n) = n^2$

$c = \log_3 7 = 1.7712$

$n^c = n^{1.7712} < n^2$

$\Rightarrow T(n) = \Theta(n^2)$

22) $T(n) = T(n/2) + n(2 - \cos x)$

$a = 1,\ b = 2,\ f(n) = 2 - \cos x$

$c = \log_2 1 = 0$

$n^c = n^0 = 1$

$\Rightarrow n^c < n(2 - \cos x)$

$T(n) = \Theta(n(2 - \cos x))$