Homework – 1

(Solution)

Homework for Notes 1 – 4

1. 1.4

id Array

		0	**1**	**2**	**3**	**4**	**5**	**6**	**No. of times id array accessed only in for loop**
P	**Q**								
0	2	2	1	2	3	4	5	6	(1)
1	4	2	4	2	3	4	5	6	(1)
2	5	5	4	5	3	4	5	6	(2)
3	6	5	4	5	6	4	5	6	(1)
0	4	4	4	4	6	4	4	6	(3)
6	0	4	4	4	0	4	4	0	(2)
1	3	4	4	4	0	4	4	0	(0)

---

2. 1.5

id Array

		0	**1**	**2**	**3**	**4**	**5**	**6**	**No. of times id array accessed only in for loop**
P	**Q**								
0	2	2	1	2	3	4	5	6	(1)
1	4	2	4	2	3	4	5	6	(1)
2	5	2	4	5	3	4	5	6	(1)
3	6	2	4	5	6	4	5	6	(1)
0	4	2	4	5	6	4	4	6	(1)
6	0	2	4	5	6	4	4	5	(1)
1	3	2	4	5	6	5	4	5	(1)

3. 1.7

		id Array							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**No. of times id array accessed only in for loop**
P	**Q**								
0	2	0	1	0	3	4	5	6	(1)
1	4	0	1	0	3	1	5	6	(1)
2	5	0	1	0	3	1	0	6	(1)
3	6	0	1	0	3	1	0	3	(1)
0	4	0	0	0	3	1	0	3	(1)
6	0	0	0	0	0	1	0	3	(1)
1	3	0	0	0	0	1	0	3	(1)

4. 1.8

		id Array							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**No. of times id array accessed only in for loop**
P	**Q**								
0	2	0	1	0	3	4	5	6	(1)
1	4	0	1	0	3	1	5	6	(1)
2	5	0	1	0	3	1	0	6	(1)
3	6	0	1	0	3	1	0	3	(1)
0	4	0	0	0	3	1	0	3	(1)
6	0	0	0	0	0	1	0	3	(1)
1	3	0	0	0	0	1	0	3	(1)

5. 1.10

For this problem N = 10^9

And no. of p-q combinations = 10^6

For ease of calculation we put a number against each of the line of the program 1.1

```
Public Class QuickF
Line no.        { public static void main (String [] args)
1.              { int N = Integer.parseInt(args [0]);
2.              int id [] = new int [N];
3.              for (int i=0; i<N; i++) id[i] = i;
4.              for (In.init() ; !In.Empty(); )
5.              { int p = In.getInt(); int q = In.getInt();
6.              int t = id[p];
7.              if (t == id[q]) contine ;
8.              for (int i=0; i<N; i++)
9.              if ( id[i] == t)  id[i] = id[q];
10.             Out.println( " " + p + " " + q );
                        }
                }
}
```

Also we assume - the no of instructions in each of the 10 lines

Line No.	No. of instructions
1	1
2	1
3	4 x N = 10^9 (each iteration 4 instr. in exec.)
4	it runs 10^6 times in case of 4-10 loop
5	2

6	1
7	1
8	per iteration consists of 10 instructions only
9	1
10	

So with $N = 10^9$ and no. of p-q pairs 10^6

Total number of instructions are :

$1 + 1 + 4x10^9 + 10^6\{2 + 1 + 1 + 10^9x10 + 1\}$

$= 1 + 1 + 4x10^9 + 10^6\{5 + 10^{10}\}$

$= 2 + 4x10^9 + 5 \times 10^6 + 10^{16}$

$= 2 + 10^6\{ 4 \times 10^3 + 5 + 10^{10}\}$

$= 2 + 10^6\{ 4000 + 5 + 10^{10}\}$

$= 2 + 10^6\{ 4005 + 10^{10}\}$

= 10000004005000002 = X

This number of instructions solely depends upon the assumption in the previous page.

Total time needed to execute X number of instructions is = $X / 10^9$ sec

= 10000004.005000002 sec

= 10000004.005000002 days / 3600 X 24

= 115.74078709 days. **(Ans)**

This is the minimum amount of time, since we have taken 10 as the number of instructions the inner “for loop” This is minimum, since we have combined 10 in each case.

6. 1.11

For this problem N = 10^9

And no. of p-q combinations = 10^6

For ease of calculation we put a number against each of the line of the program:

Public Class QuickUW

Line no.

1. { public static void main (String [] args)
2. { int N = Integer.parseInt(args [0]);
3. int id [] = new int [N], sz [] = new int [N];
4. for (int i=0; i<N; i++)
5. {id[i] = i; sz[i] = 1;}
6. for (In.init() ; !In.Empty();)
7. { int i, j, p = In.getInt(), q = In.getInt();
8. for (i=p; i != id[i]; i = id[i]) ;
9. for (j=q; j != id[j]; j = id[j]) ;
10. if (i == j) continue;
11. if (sz[i] < sz[j])
12. { id[i] = j; sz[j] += sz[i];}
13. else
14. { id[j] = i; sz[i] += sz[j];}
15. Out.println(" " + p + " " + q);

```
        }
    }
}
```

Now we assume the number of instructions in each of these lines are as follows:

Line No.	No. of instructions
2	1
3	2
4	{line number 4 + 5 totally executes
5	$2 \times N = 2 \times 10^9$ }
6	{ the outer for loop works from
‘	6 – 15 and it is specified that each iterations takes
‘	100 instructions (at max). the no. 6 – 15 executes:
15	= 100 x (no. of input pairs)
	= 100×10^6

Total number of instructions:

$1 + 2 + 2 \times 10^9 + 100 \times 10^6$

$= 3 + 2 \times 10^9 + 10^8$

$= 3 + 10^8(20+1) = 3 + 10^8(21) = 2100000000 + 3 = 2100000003$

= max execution time = $2100000003 / 10^9$ sec. 2.100000003 sec. **(Ans)**

---

7. 6.20 **Selection Sort**

Input	->	E	A	S	Y	Q	U	E	S	T	I	O	N
Iteration (1) no.	->	A	E	S	Y	Q	U	E	S	T	I	O	N

(2)	->	A	E	S	Y	Q	U	E	S	T	I	O	N
(3)	->	A	E	E	Y	Q	U	S	S	T	I	O	N
(4)	->	A	E	E	I	Q	U	S	S	T	Y	O	N
(5)	->	A	E	E	I	N	U	S	S	T	Y	O	Q
(6)	->	A	E	E	I	N	O	S	S	T	Y	U	Q
(7)	->	A	E	E	I	N	O	Q	S	T	Y	U	S
(8)	->	A	E	E	I	N	O	Q	S	T	Y	U	S
(9)	->	A	E	E	I	N	O	Q	S	S	Y	U	T
(10)	->	A	E	E	I	N	O	Q	S	S	T	U	Y
(11)	->	A	E	E	I	N	O	Q	S	S	T	U	Y
(12)	->	A	E	E	I	N	O	Q	S	S	T	U	Y

Answer.

---

8. 6.24

Insertion Sort

Input String	E	A	S	Y	Q	U	E	S	T	I	O	N
	A	E	S	Y	Q	U	E	S	T	I	O	N
	A	E	S	Y	Q	U	E	S	T	I	O	N
	A	E	S	Y	Q	U	E	S	T	I	O	N
	A	E	Q	S	Y	U	E	S	T	I	O	N
	A	E	Q	S	U	Y	E	S	T	I	O	N
	A	E	E	Q	S	U	Y	S	T	I	O	N
	A	E	E	Q	S	S	U	Y	T	I	O	N
	A	E	E	Q	S	S	T	U	Y	I	O	N
	A	E	E	I	Q	S	S	T	U	Y	O	N
	A	E	E	I	O	Q	S	S	T	U	Y	N
	A	E	E	I	N	O	Q	S	S	T	U	Y
	A	E	E	I	N	O	Q	S	S	T	U	Y

Answer.

---

9. 8.5

Input String - A E Q S U Y E I N O S T

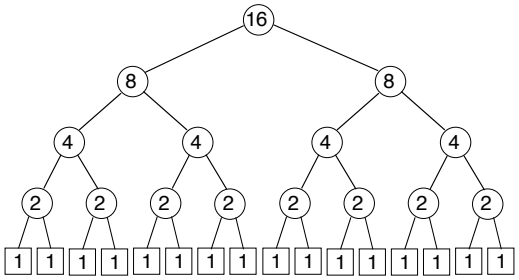
```
A E Q S U Y E I N O S T
  E Q S U Y E I N O S T A
    Q S U Y E I N O S T A E
```

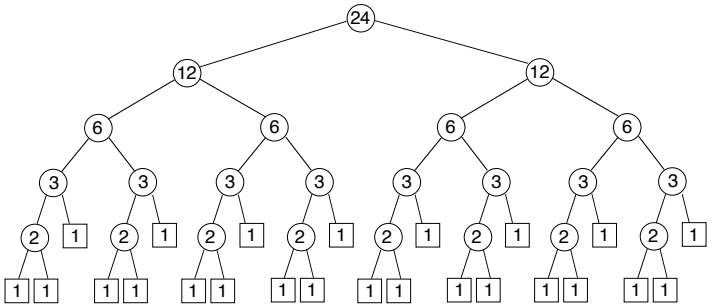
```
Q  S  U  Y     I  N  O  S  T  A  E  E
Q  S  U  Y        N  O  S  T  A  E  E  I
Q  S  U  Y           O  S  T  A  E  E  I  N
Q  S  U  Y              S  T  A  E  E  I  N  O
   S  U  Y              S  T  A  E  E  I  N  O  Q
      U  Y              S  T  A  E  E  I  N  O  Q  S
      U  Y                 T  A  E  E  I  N  O  Q  S  S
      U  Y                    A  E  E  I  N  O  Q  S  S  T
         Y                    A  E  E  I  N  O  Q  S  S  T  U
                              A  E  E  I  N  O  Q  S  S  T  U  Y
```

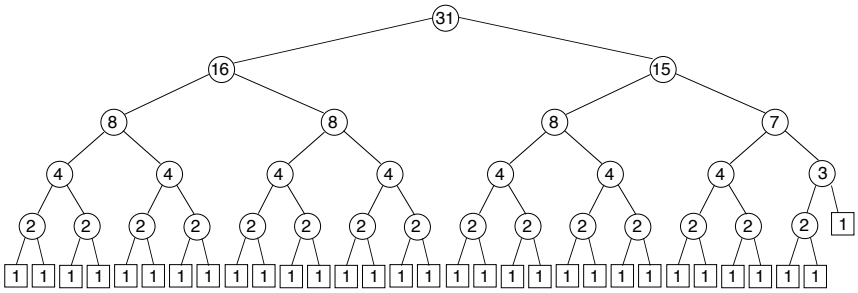
10. 8.10

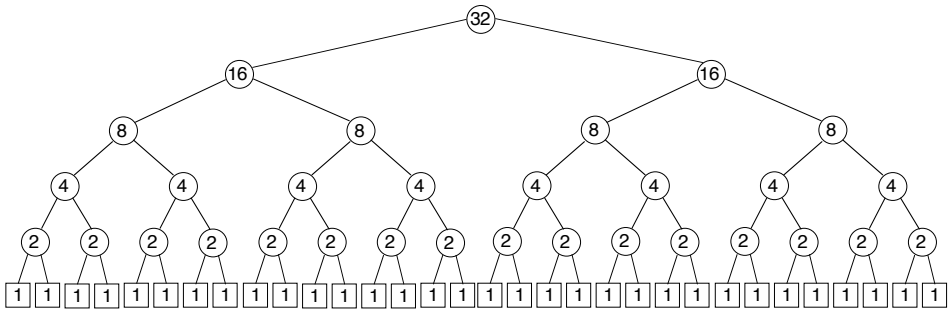
Divide and conquer trees for

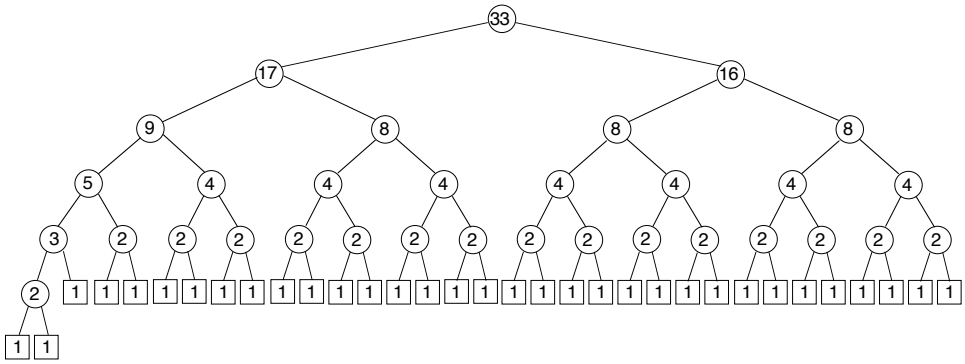
N = 16 , 24, 31, 32, 33, & 39

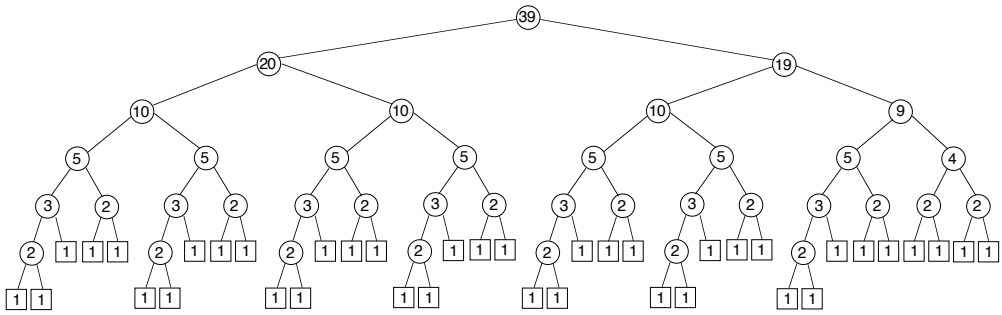








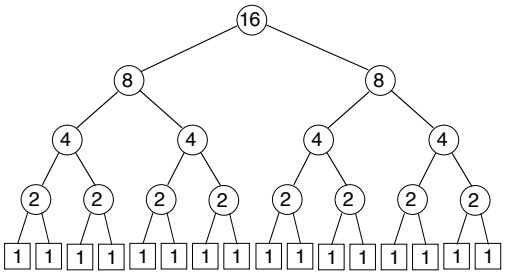


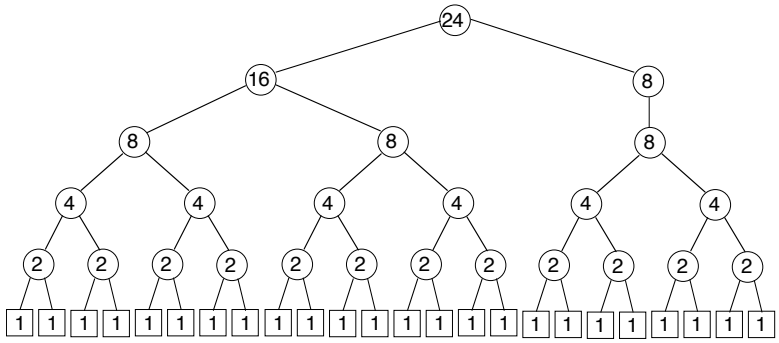


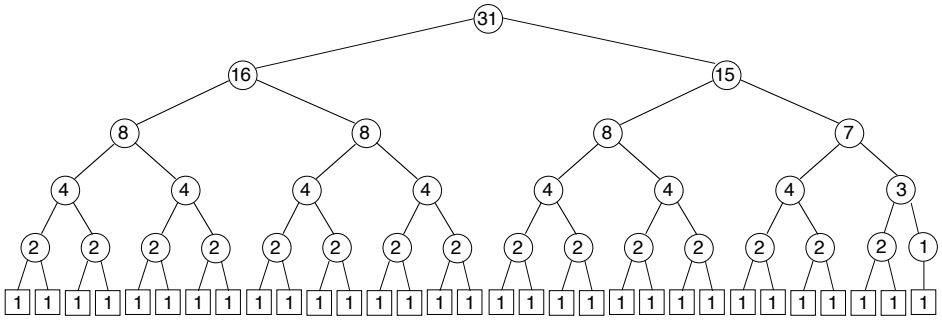
---

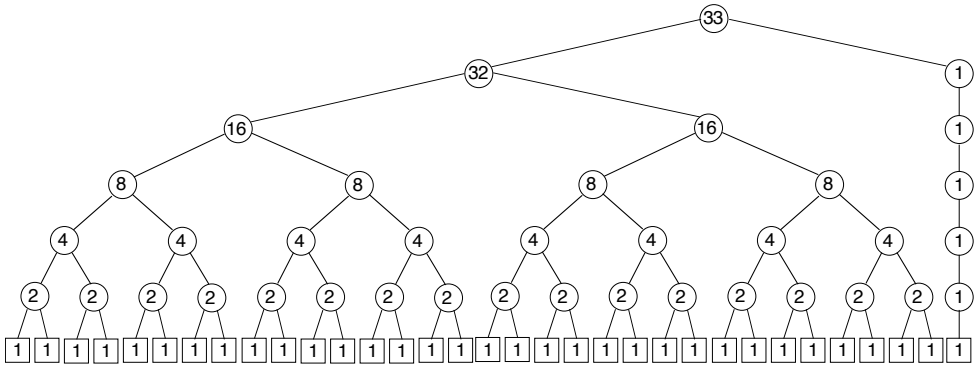
11. 8.26

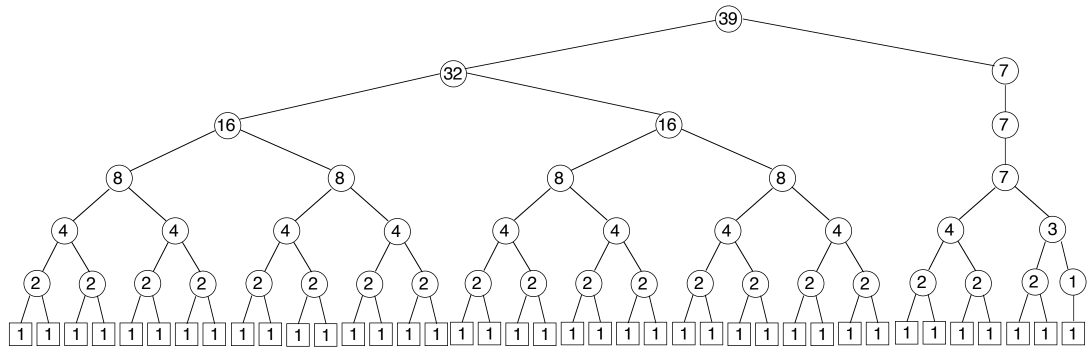
N = 16 , 24, 31, 32, 33, & 39


16
8 8
4 4 4 4
2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1


24
16 8
8 8 8
4 4 4 4 4 4
2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1


31
16 15
8 8 8 7
4 4 4 4 4 4 4 3
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1


33
32 1
16 16 1
8 8 8 8 1
4 4 4 4 4 4 4 4 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1



12. Write a function to determine (in logarithmic time) the range of elements in a strictly increasing sequence $a_0, a_1, a_2, \ldots, a_{n-1}$ that have $a_i = i$.

```
// Binary search for some element with a[i]==i
low=0;
high=n-1;
while (1)
{
  if (high<low)
  {
    printf("Range is empty, bracketed by %d %d\n",high,low);
    return;
  }
  mid=(low+high)/2;
  if (a[mid]==mid)
    break;
  if (a[mid]<mid)
    low=mid+1;
  else
    high=mid-1;
}
// Find beginning of range
low1=low;
high1=mid;
while (low1<=high1)
{
  mid1=(low1+high1)/2;
  if (a[mid1]==mid1)
    high1=mid1-1;
  else
    low1=mid1+1;
}
printf("%d begins the range\n",low1);
// Find end of range
low2=mid;
high2=high;
while (low2<=high2)
{
  mid2=(low2+high2)/2;
  if (a[mid2]==mid2)
    low2=mid2+1;
  else
    high2=mid2-1;
}
printf("%d ends the range\n",high2);
```

13. 2.5

$10\ N \lg N > 2N^2$

or, $\frac{10\ N \lg N}{2N^2} > 1$ or, $\frac{5\ \lg N}{N} > 1$ or, $\frac{\lg N}{N} > 1/5$

or, $5\ lg_2 N > N$ ----- this inequality holds for all

N = 1,2,3 ….. . .. 22

(Ans.)

$10\ N\ lgN > 2\ N^2$

For N = 1,2,3 ….. . .. 22

---

14. 2.8

$log_{10}\ log_{10}\ N > 8$

The equality condition is

$log_{10}\ log_{10}\ N = 8$

$log_{10}\ (X) = 8$ (say , $log_{10}\ N = X$)

X = 10^8 Now $log_{10}\ N = X$

or, $log_{10}\ N = 10^8$

or, N = 10^{10^8} ------ N being an integer, the smallest value for which $log_{10}\ log_{10}\ N > 8$ is

$10^{10^8} + 1$ **(Ans.)**

---

15. 2.15

lg (N!)

using Sterling's approximation,

N! = O (N^N)

Lg (N!) = lg (N^N)

= N lg (N)

To represent lg (N!) = lg (N lg N) + 1 bits are needed **(Answer.)**

---

16. 2.21

f (N) -> O (f (N))

from the definition $c_0 = 1$ and $N = N_0$,

f (N) -> O (f (N))

---

c . O (f (N)) - > O (f (N))

N being large

c . O (f (N)) - > O (f (N)) [from the definition with c = 1]

c . O (f (N)) - > O (f (N)) **(Answer)**

O (c . f (N)) - > O (f (N))

O (c . f (N)) = c (O (f(N))) [from the definition with c = 1]

= O (f(N)) **(Answer)**

---

f (N) – g (N) = O (h (N))

or, f (N) = g (N) + O (h (N)) [since this is symmetric]

O (f (N)) O (g (N))

= O (f(N) g (N)) [from big-Oh property]

O (f (N)) + O (g (N))

= O (g (N)) + O (g (N)) if f(N) = O (g (N))

= 2 . O (g (N))

= O (g (N)) [since c . O (f (N)) = O (f(n))]

Proved

---

17. 2.25

$$\frac{N}{N+O(1)} = 1 + O\left(\frac{1}{N}\right)$$

f (N) = $\frac{N}{N+O(1)}$

g (N) = $1 + O\left(\frac{1}{N}\right)$

$$lim_{N\to\infty} \frac{f(N)}{g(N)} = lim_{N\to\infty} \frac{\frac{N}{N+O(1)}}{1+O\left(\frac{1}{N}\right)}$$

$$= lim_{N\to\infty} \frac{N}{(N+O(1))(1+O\left(\frac{1}{N}\right))}$$

$$= lim_{N\to\infty} \frac{N}{O(N)(1+O\left(\frac{1}{N}\right))}$$ --- N+O(1) = O(N)

$$= lim_{N\to\infty} \frac{O(N)}{O(N)(1+O\left(\frac{1}{N}\right))}$$

$$= lim_{N\to\infty} \frac{1}{1+O\left(\frac{1}{N}\right)}$$

$$\cong C \ (constant)$$

f (N) = O (g(N))

$\frac{N}{N+O(1)} = 1 + O\left(\frac{1}{N}\right)$ **(proved)**

18.

O (n^2) Proof

Assume $\sum_{i=1}^{n} i \leq cn^2$

$$\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + n + 1 \leq cn^2 + n + 1 = c(n+1)^2 - 2cn - c + n + 1$$

$$\leq c(n+1)^2 for\ n \geq -\left(\frac{c-1}{2c-1}\right) i.e\ 1 \leq c < \infty, n \geq 0$$

Ω (n^2) Proof

Assume $\sum_{i=1}^{n} i \geq cn^2$

$$\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + n + 1 \geq cn^2 + n + 1 = c(n+1)^2 - 2cn - c + n + 1$$

$$\geq c(n+1)^2 for\ c \leq \left(\frac{n+1}{2n+1}\right)$$

$$i.e\ c \leq \frac{1}{2} + \frac{1}{2(2n+1)} => \infty c \leq \frac{1}{2}, n \geq 0$$

19.

T(n) = 2T($\frac{n}{4}$) + $\sqrt{n}$

O ($\sqrt{n}$ log n) proof.

Assume $T(k) \leq c\sqrt{k}\log k$

$$T\left(\frac{n}{4}\right) \leq c\sqrt{\frac{n}{4}}\log\frac{n}{4} = \frac{c\sqrt{n}}{2}\log n - \frac{c\sqrt{n}}{2}\log 4$$

$$T(n) \leq c\sqrt{n}\log n - c\sqrt{n}\log 4 + \sqrt{n}$$

$= c\sqrt{n}\log n - c\,(\log 4 - 1)\sqrt{n}$

$\leq c\sqrt{n}\log n \quad for\ -1 + clog\ 4 \geq 0$

⇨ c log 4 ≥ 1

⇨ c ≥ $\frac{1}{\log 4} = \frac{1}{2}$

Ω (√n log n) . proof.

Assume $T(k) \geq c\sqrt{k}\log k$

$T\left(\frac{n}{4}\right) \geq \frac{c}{2}\sqrt{n}\log n - \frac{c}{2}\sqrt{n}\log 4$

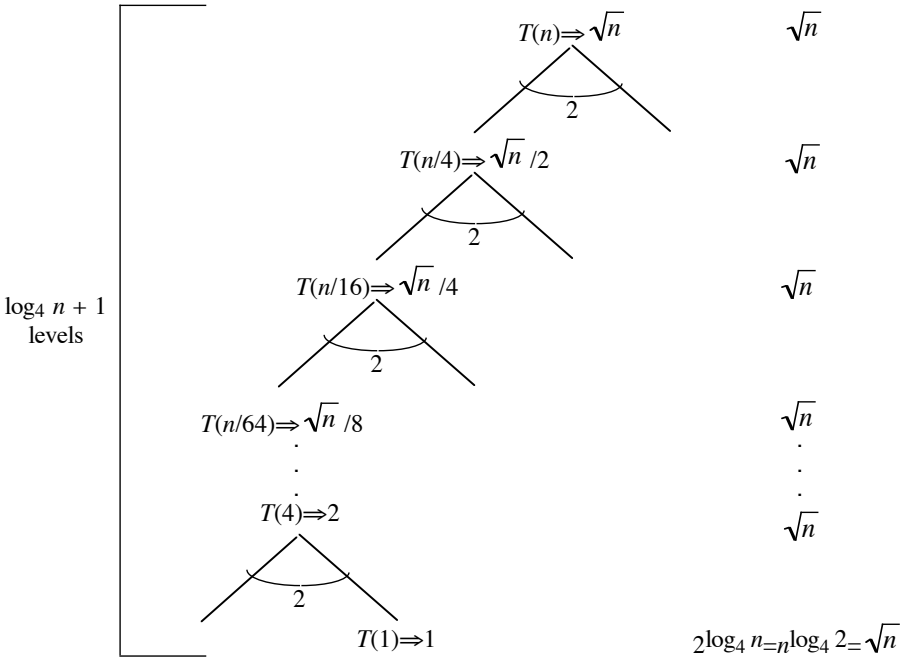
$T(n) \geq c\sqrt{n}\log n - c\sqrt{n}\log 4 + \sqrt{n}$

$= c\sqrt{n}\log n + \sqrt{n}\,(1 - c\log 4)$

$\geq c\sqrt{n}\log n \quad for\ clog\ 4 \leq 1$

⇨ 0 <c ≤ $\frac{1}{log4}$

⇨ 0 <c ≤ $\frac{1}{2}$



No. of levels = log $\sqrt{n} + 1$

$$\sum_{k=0}^{lg\sqrt{n}} \sqrt{n} = \left(\frac{lgn}{2} + 1 \right)\sqrt{n} = \frac{\sqrt{n}\lg n}{2} + \sqrt{n} = O(\sqrt{n}\lg n)$$

20.

T(n) = 3T($\frac{n}{3}$) + n^2

O (n^2) . Proof.

Assume $T(k) \leq ck^2$

$T\left(\frac{n}{3}\right) \leq \frac{cn^2}{9}$ => $T(n) \leq 3\frac{cn^2}{9} + n^2$

$= \frac{cn^2}{3} + n^2$ => $cn^2 - \frac{2}{3}cn^2 + n^2$

$\leq cn^2 \; if \; c \geq \frac{3}{2}$

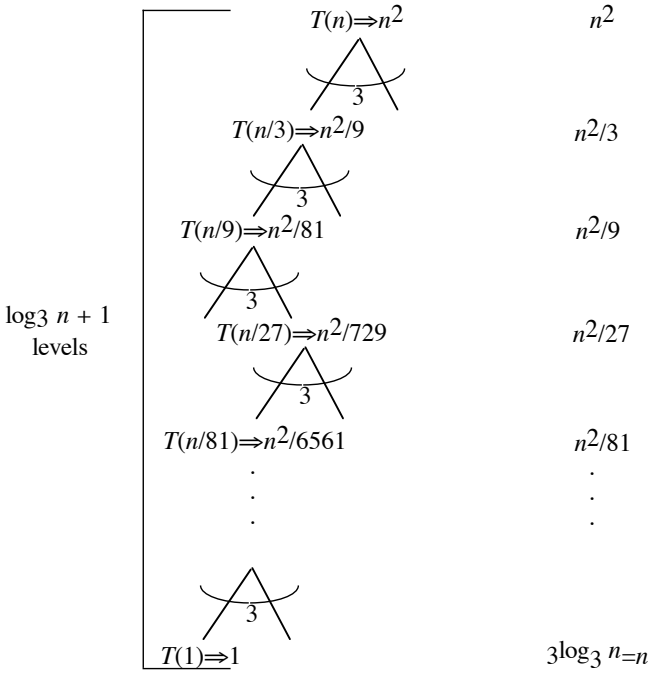
Ω (n^2) . Proof.

Assume $T(k) \geq ck^2$

$T\left(\frac{n}{3}\right) \geq \frac{cn^2}{9}$ => $T(n) \geq 3\frac{cn^2}{9} + n^2$

$= \frac{cn^2}{3} + n^2 = cn^2 - \frac{2}{3}cn^2 + n^2$

$\geq cn^2 \; if \; 0 < c \leq \frac{3}{2}$



Using indefinite geometric sum formula:

$$n^2 \sum_{k=0}^{\log_3 n - 1} \frac{1}{3^k} + n \leq n^2 \sum_{k=0}^{\infty} \frac{1}{3^k} + n$$

$$= n^2 \frac{1}{1 - \frac{1}{3}} + n \qquad \text{From } \sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \qquad 0 < x < 1$$

$$= \frac{3}{2}n^2 + n = O(n^2)$$

21.

T(n) = 2T($\frac{n}{4}$) + 1

O ($\sqrt{n}$) . Proof.

Assume $T(k) \leq c\sqrt{k}$

$T\left(\frac{n}{4}\right) \leq \frac{c}{2}\sqrt{n}$ => $T(n) \leq c\sqrt{n} + 1$ stuck !!!!!

Examine a few cases ……. :-

T(1) = d

T(4) = 2T(1) + 1 = 2d + 1

T(16) = 2T(4) + 1 = 4d + 3

.

.

T(n) = $\sqrt{n}\,d$ + ($\sqrt{n}$ - 1) = (d + 1) $\sqrt{n} - 1 = c\sqrt{n} - 1$ $where \quad c = d + 1$

Assume

$T(K) \leq c\sqrt{k} - 1$

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

$\leq c\sqrt{n} \; for \; 0 < c < \infty$

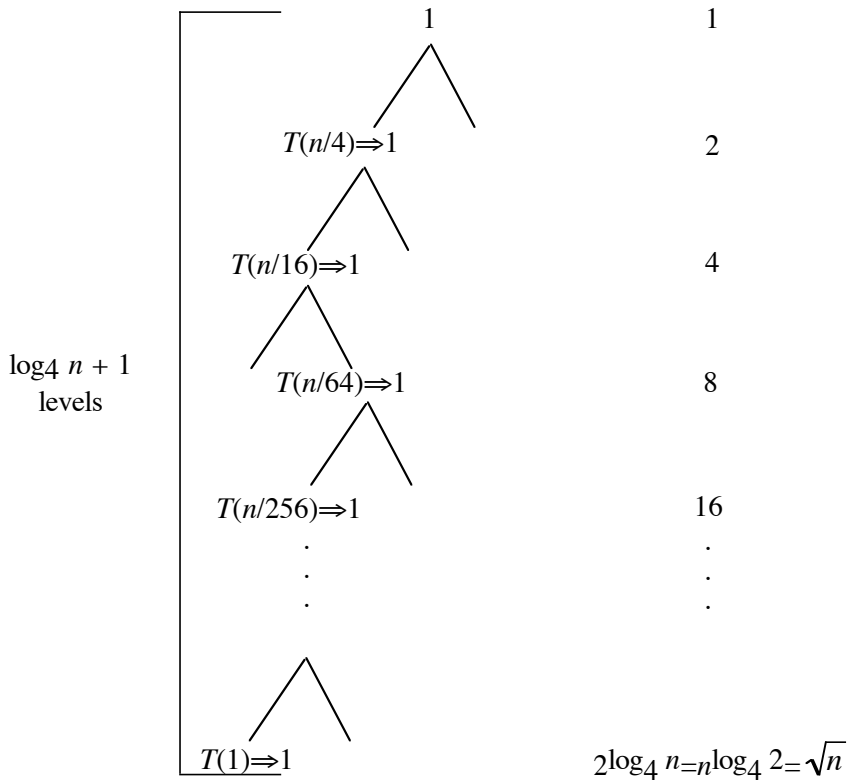
Ω($\sqrt{n}$). Proof.

$T(K) \geq c\sqrt{k}$

$T\left(\frac{n}{4}\right) \geq \frac{c}{2}\sqrt{k}$

$T(n) \geq c\sqrt{n} + 1$

$\geq c\sqrt{n} \quad for \; 0 < c < \infty$



Using definite geometric sum formula:

$$\sum_{k=0}^{\log_4 n-1} 2^k + \sqrt{n} = \frac{2^{\log_4 n} - 1}{2-1} + \sqrt{n} \qquad \text{Using } \sum_{k=0}^{t} x^k = \frac{x^{t+1}-1}{x-1} \qquad x \neq 1$$

$$= n^{\log_4 2} - 1 + \sqrt{n}$$

$$= 2\sqrt{n} - 1 = \Theta\left(\sqrt{n}\right)$$