HW-1

1.4

P	Q
0	2
1	4
2	5
3	6
0	4
6	0
1	3

id Array

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

1.5 Ans:

P	Q
0	2
1	4
2	5
3	6
0	4
6	0
1	3

id array

0	1	2	3	4	5	6	
2	1	2	3	4	5	6	(1)
2	4	2	3	4	5	6	(1)
2	4	5	3	4	5	6	(1)
2	4	5	6	4	5	6	(1)
2	4	5	6	4	4	6	(1)
2	4	5	6	4	4	5	(1)
2	4	5	6	5	4	5	(1)
2	4						

1.7.

p	q
0	2
1	4
2	5
3	6
0	4
6	0
1	3

id array

0	1	2	3	4	5	6	
0	1	0	3	4	5	6	(1)
0	1	0	3	1	5	6	(1)
0	1	0	3	1	0	6	(1)
0	1	0	3	1	0	3	(1)
0	0	0	3	1	0	3	(1)
0	0	0	0	1	0	3	(1)
0	0	0	0	1	0	3	(1)

1.8.

p	q
0	2
1	4
2	5
3	6
0	4
6	0
1	3

id array

0	1	2	3	4	5	6	
0	1	0	3	4	5	6	(1)
0	1	0	3	1	5	6	(1)
0	1	0	3	1	0	6	(1)
0	1	0	3	1	0	3	(1)
0	0	0	3	1	0	3	(1)
0	0	0	0	1	0	3	(1)
0	0	0	0	1	0	3	(1)

1.10. For this problem $N = 10^9$

and no of p-q combination $= 10^6$.

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

Line no

```
    public class QuickF
    { Public static void main( String [ ] args ).
      { int   N = Integer. parseInt (args [0]);
1.      int id[] = new int [N];
2.  for (int  i = 0; i < N; i++) id [i] = i;
3.      for (In.init(); !In.empty(); )
4.      { int p = In.getInt(); int q = In.getInt();
5.        int t = id [p];
6.        if (t = id [q]) continue;
7.        for (int i = 0; i<N; i++)
8.          if (id [i] == t) id [i] = id [q];
9.      Out.println(" " + p + " " + q);
10.
      }
    }
}
```

Also, we assume – The no of instructions in each of 10 lines in

Line no	No of instructions
1.	1
2.	1
3.	$4 \times N = 10^9$ sec (each iteration 4 ins is executed)
4.	It runs 10^6 times in the case 4–10 loop.
5.	2
6.	1
7.	1
8.	2 per iteration consists of 2 instructions only
9.	
10.	1

So, here with $N = 10^9$ & no of p-q pair 10^6,

Total no of ins.

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

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

$$=2+4*10^9+5*10^6+10^{16}$$

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

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

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

$$=10000004005000002 = X$$

This no of instructions solely depends on the assumptions in the previous page.

∴ Total time needed to execute the X no of instructions $= X/10^9$ sec.

$$= 10000004.005000002 \text{ sec}$$

$$= \frac{10000004.005000002}{3600 \times 24} \text{ days}$$

$$= 115.74078709 \text{ days. (Ans).}$$

This is the minimum amount of time since, we have taken to do the no of instructions for the inner "for" loop. This is min^m since we have considered 1 in each case.

1.11.

N here = 10^9

no of p-q pairs = 10^6.

We put line numbers against each of the program line as follows:—

```
Lines
1.  public class Quick0
    {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 no of instructions in each of these lines are as follows.

Line no	No of Ins.
2	1.
3.	2.
4.	{ line no 4 & 5 totally executes $2 \times N = 2 \times 10^9$ time }
5	
6	The outer for loop works from 6–15. and it is specified that each iteration takes 100 instructions (at max^{m}).
⋮	∴ line no 6 – 15 executes $= 100 \times$ {no of input pair} $= 100 \times 10^6$.
15	

∴ Total no 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^{m} execution time $= \frac{2100000003}{10^9}$ sec.

$= 2.100000003$ (Ans)

6.20 Selection Sort:-

Input → E A S Y Q U E S T I O N

Iteration no(1) → 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).

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 Q S U~~

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 I 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)

8.5 INPUT STRING - AE@SUYEINOST

AE@SUYEINOST

AE@SUYEINOST A

E@SUYEINOST A

@SUYEINOST AE

@SUYINOST AEE

@SUY NOST AEEI

@SUY OST AEEIN

@SUY ST AEEINO

SUY ST AEEINO@

UY ST AEEINO@S

UY T AEEINO@SS

UY AEEINO@SST

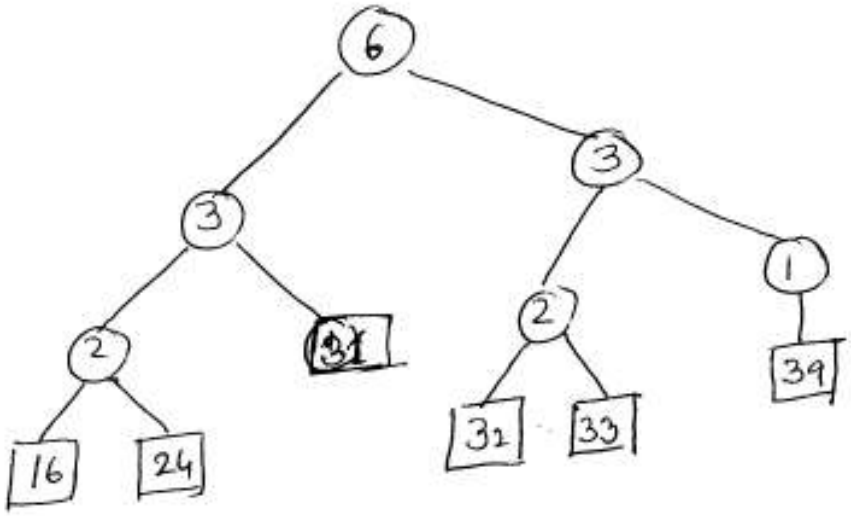
Y AEEINO@SSTU

AEEINO@SSTUY

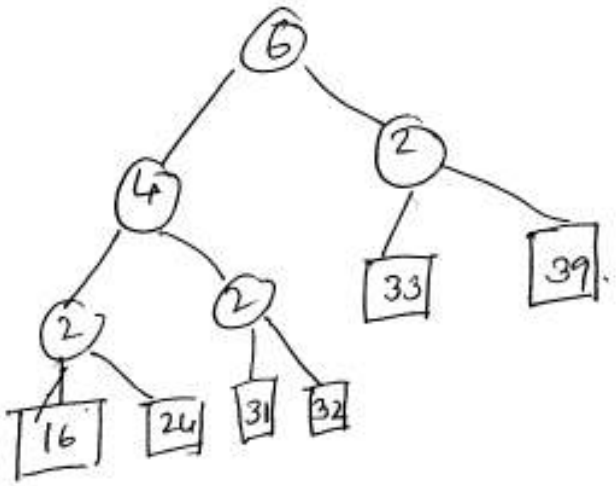
8.10.

Divide-and-Conquer Trees for

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



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



12. As the nos are strictly increasing, so we assume,

$$a_i > a_j \quad \forall (i > j)$$

Algo^m

```
start = 0; end = n-2;
if a[0] ≠ 0 then
print{"no strictly increasing subsequence exists"}
return;
if a[n] = n then
print{" the largest strictly increasing subsequence
        is a0 ... an-1}
return;
while (start ≠ end)
{ mid = (start + end) /2;
  if a[mid] ≠ mid then
      end = mid-1;
  else
      start = mid+1;
}
print{" the largest strictly
      increasing subsequence is
        a0 ... ... end"}.
return;
```

2.5. $10N \lg N > 2N^2$

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

or, $\frac{5 \lg N}{N} > 1$

or, $\frac{\lg N}{N} > 1/5$.

or, $5 \log_2 N > N$.

or, $5 \lg_2 N > N$.

This inequation holds for all

$N = 1, 2, 3, \ldots, 22$.

(Ans)

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

for $N = 1, 2, 3, \ldots, 22$.

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

The equality condⁿ is

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

or, $\log_{10}(X) = 8$ $\left(\text{say, } \log_{10} N = X\right)$.

or, $X = 10^8$.

Now, $\log_{10} N = X$.

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

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

$\therefore$ N being an integer, the smallest value for which $\log_{10}\log_{10} N > 8$ is $\left(10^{10^8} + 1\right)$. (Ans).

2.15 : $\lg(N!)$.

Using Stirling's approximation,

$$N! = O(N^N).$$

$$\therefore \lg(N!) = \lg(N)^N$$

$$= N \lg N.$$

$\therefore$ To represent $\lg(N!) = \lg(N \lg N) + 1$ bits are needed (Ans).

2.21 $f(N) \to O(f(N))$.

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

$f(N) \to O(f(N))$.

$C\,O(f(N)) \to O(f(N))$

N being large.

$C \cdot O(f(N)) \Rightarrow O(f(N))$ [from the definition with $C=1$].

$\therefore C\,O(f(N)) \to O(f(N))$ (Ans)

$O(Cf(N)) \to O(f(N))$

$O(Cf(N)) = C(O(f(N)))$. [from the definition $C=1$].

$= O(f(N))$ (Ans).

$f(N) - g(N) = O(h(N))$ or

or, $f(N) = g(N) + O(h(N))$.

[since, ~~f(N)~~ 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 $f(N) = O(g(n))$

$= 2 \cdot O(g(N))$

$= O(g(N))$ [since $c \cdot O(f(n)) = O(f(n))$]

(proved).

2.25 $\dfrac{N}{N+O(1)} = 1 + O(1/N)$.

$\therefore f(N) = N/N+O(1)$

$g(N) = 1 + O(1/N)$.

$\therefore \lim_{N\to\infty} \dfrac{f(N)}{g(N)} = \lim_{N\to\infty} \dfrac{\frac{N}{N+O(1)}}{1+O(1/N)}$

$= \lim_{N\to\infty} \dfrac{N}{(N+O(1))(1+O(1/N))}$

$= \lim_{N\to\infty} \dfrac{N}{O(N)(1+O(1/N))}$ $\left[\because N+O(1) = O(N)\right]$

$= \lim_{N\to\infty} \dfrac{O(N)}{O(N)(1+O(1/N))}$

$= \dfrac{1}{1+O(1/N)}$.

$\cong C$ (constant)

$\therefore f(N) = O(g(N))$

$\therefore \dfrac{N}{N+O(1)} = 1 + O(1/N)$ (proved).

HW 1-4

18) $O(n^2)$ Proof.

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

$$\therefore \sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + n+1 \le cn^2 + n + 1 = c(n+1)^2 - 2cn - c + n + 1$$
$$\le c(n+1)^2 \text{ for } n \ge -\left(\frac{c-1}{2c-1}\right)$$

i.e. $1 \le c < \infty$

$n \ge 0$.

$\Omega(n^2)$ Proof.

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

$$\therefore \sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + n+1 \ge cn^2 + n + 1 = c(n+1)^2 - 2cn - c + n + 1$$
$$\ge c(n+1)^2 \text{ for } c \le \frac{n+1}{2n+1}$$
$$\Rightarrow c \le \frac{1}{2} + \frac{1}{2(2n+1)}$$
$$\Rightarrow c \le 1/2 \text{ , } n \ge 0.$$

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

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

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

$$\therefore T(n/4) \le c\sqrt{n/4} \log(n/4) = \frac{c\sqrt{n}}{2} \log n - \frac{c\sqrt{n}}{2} \log 4$$

$$\therefore T(n) \le c\sqrt{n} \log n - c\sqrt{n} \log 4 + \sqrt{n}$$
$$= c\sqrt{n} \log n - (c \log 4 - 1)\sqrt{n}$$
$$\le c\sqrt{n} \log n \quad \text{for } -1 + c \log 4 \ge 0$$
$$\Rightarrow c \log 4 \ge 1$$
$$\Rightarrow c \ge \frac{1}{\log 4} = \frac{1}{2}.$$

$\Omega(\sqrt{n}\log n)$

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

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

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

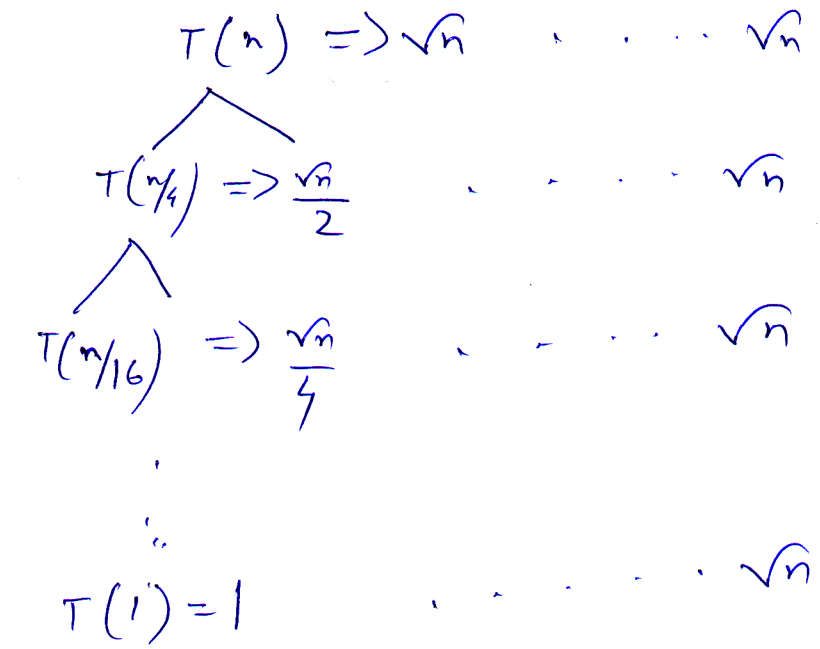
$= c\sqrt{n}\log n + \sqrt{n}(1 - c\log 4)$

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

$\Rightarrow 0 < c \leq \frac{1}{\log 4}$

or, $0 < c \leq \frac{1}{2}$.

Recursion tree :-



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

$\therefore \sum_{k=0}^{\log\sqrt{n}} \sqrt{n} = \left(\frac{\log n}{2} + 1\right)\sqrt{n} = \frac{\sqrt{n}\log n}{2} + \sqrt{n} = \Theta(\sqrt{n}\log n)$.

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

$O(n^2)$ Proof.

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

$T(n/3) \leq \frac{cn^2}{9}$.

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

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

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

$\leq cn^2$ if $c \geq 3/2$.

$\Omega(n^2)$ Proof.

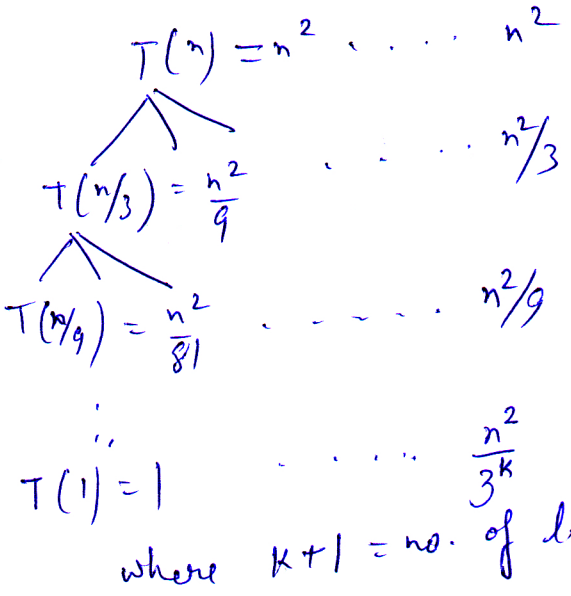
Assume $T(k) \geq ck^2$

$T(n/3) \geq \frac{cn^2}{9}$.

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

$\geq cn^2$ if $0 < c \leq 3/2$

Recursion tree :-



where $k+1$ = no. of levels.

Now, $\frac{n}{3^k} = 1 \Rightarrow k = \log_3 n$

$$\therefore \sum_{k=0}^{\log_3 n} n^2/3^k = n^2 \left(\frac{\frac{1}{3^{\log_3 n + 1}} - 1}{\frac{1}{3} - 1} \right) = \frac{3n^2}{2} - \frac{n}{2} = \Theta(n^2)$$

21) $T(n) = 2T(n/4) + 1$

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

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

$T(n/4) \leq \frac{c}{2}\sqrt{n}$

$T(n) \leq c\sqrt{n} + 1$ stuck !!

Examine a few cases :-

$T(1) = d$

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

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

$\vdots$

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

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 $0 < c < \infty$

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

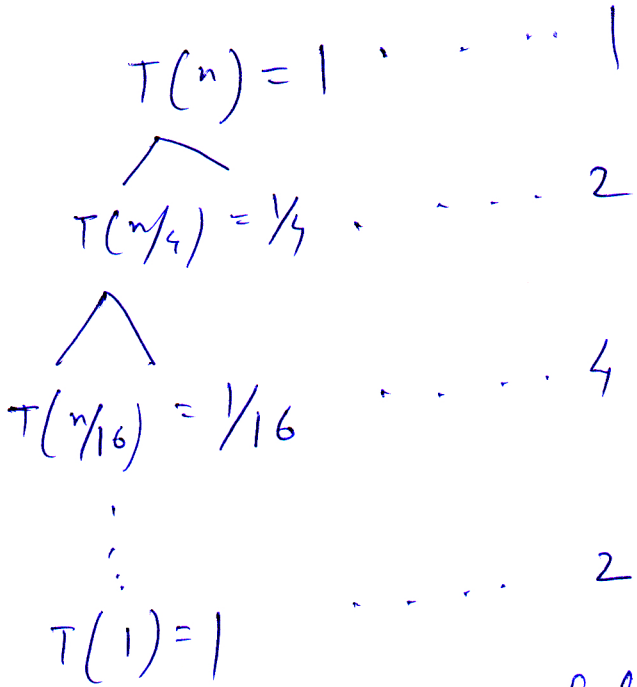
Assume, $T(k) \geq c\sqrt{k}$.

$T(n/4) \geq \frac{c}{2}\sqrt{k}$.

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

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

Recursion tree



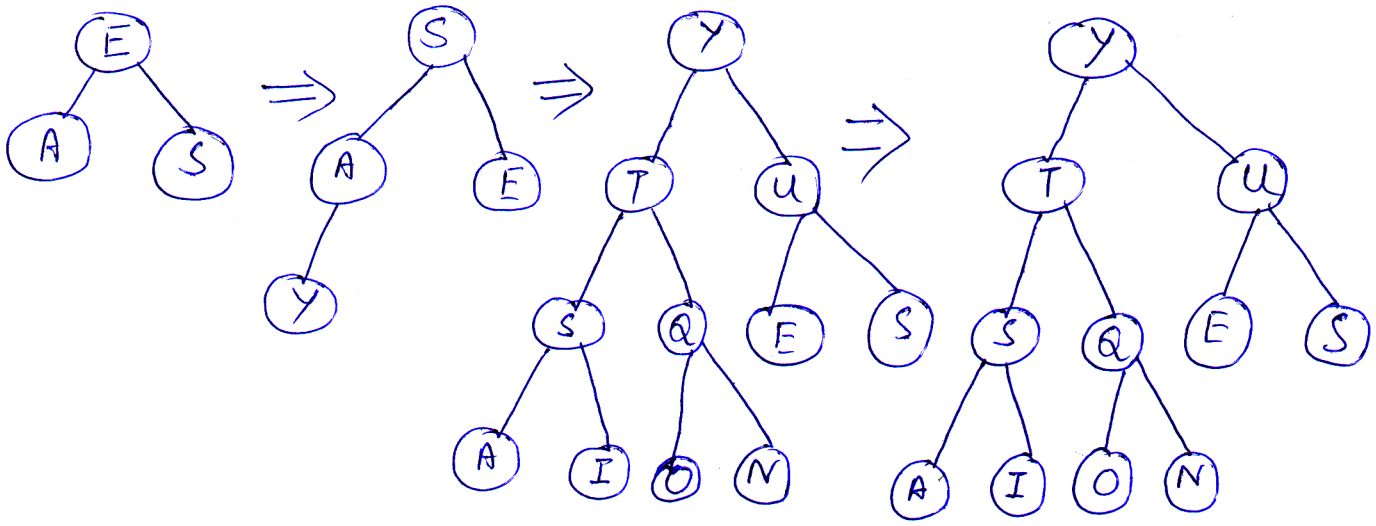
where $k+1$ = no. of levels.

and, $\frac{n}{4^k} = 1 \Rightarrow k = \log \sqrt{n}$

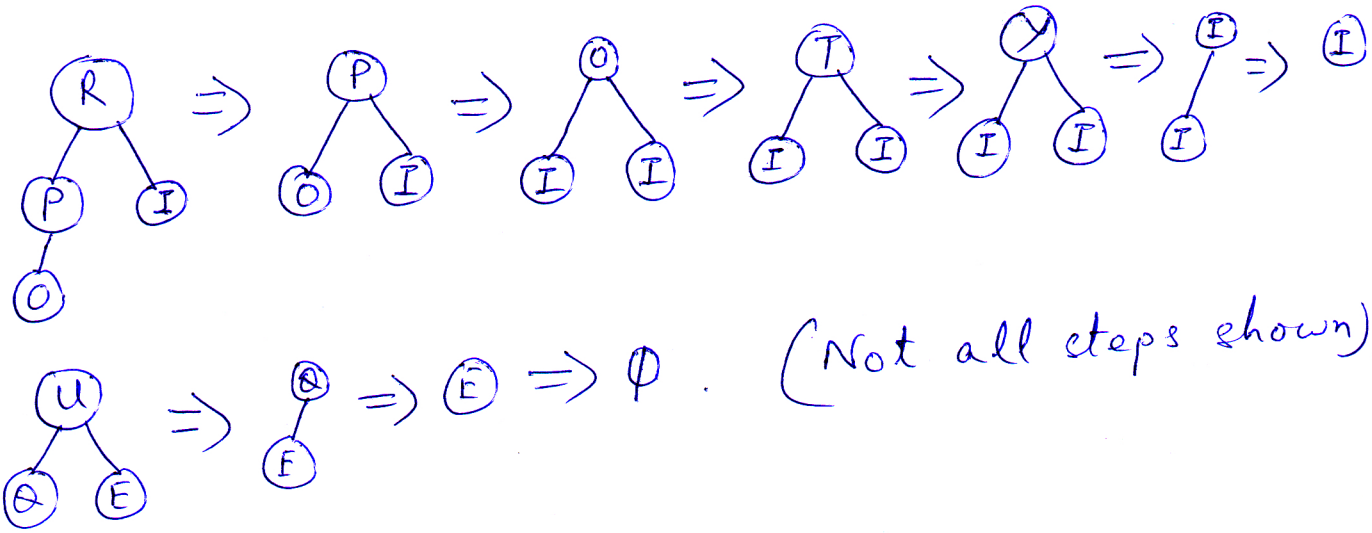
$\therefore \sum_{k=0}^{\log \sqrt{n}} 2^k = \frac{2^{\log \sqrt{n} + 1} - 1}{2-1} = 2\sqrt{n} - 1 = \theta(\sqrt{n})$.

HW 5-7

1) 9.21



2) 9.22



$\Rightarrow$ $\Rightarrow$ E $\Rightarrow \phi$. (Not all steps shown).

3) 9.28

Heapsort is not stable and cannot be made stable by an easy tie-breaking rule (as opposed to insertion sort, for example). As an example, suppose that the array elements are

5 3 3 2

which already form a heap.

First, 5 is put at the end of the array, and then one of

the two children of 5 has to become the root element, to be put into the second-to-last position. This must be the right child (i.e., the maximum child in case of a tie is to be the right child), in order to obtain a stable sorting method (otherwise, the two 3's would exchange their original order).

But now consider the original sequence

6 5 3 3 2

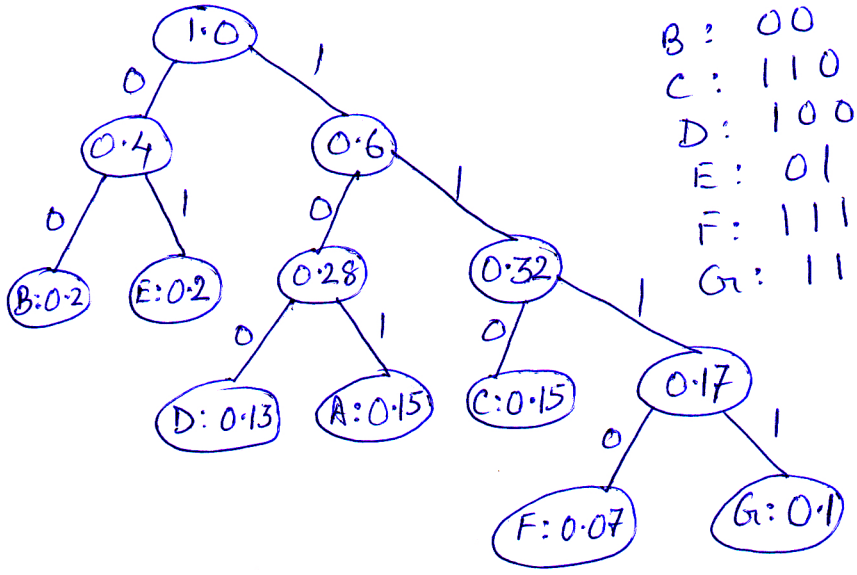
which also forms a heap. Here, the two children of 6 are 5 and 3, and the two children of 5 are the second 3 and 2. Then after extraction of 6 we get 5 at the root of the heap, and 5 is succeeded by the originally second 3. This produces the identical heap to the previous case, but the two 3's are in reverse position, so the rule that determines the maximum must violate stability one way or the other.

4.) 9.30.

```
Class Sort
{
    static void sort (ITEM [ ] a, int 1, int r)
     { HeapSort (a,1, r); }
    static void HeapSort (ITEM[ ] a, int 1, int r)
    {
        int k;
```

```
PQ pq = new PQ(r-1+1);
for (k = r/2 ; k>=1 ; k--)
pq.sink (k, r);
while (r>1)
  {pq.exch (1, r) ;pq.sink (1, --r);}
}
}
```

5>



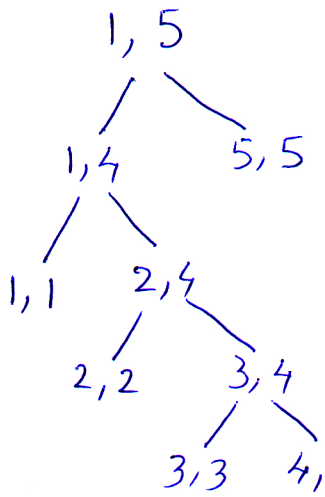
A : 101
B : 00
C : 110
D : 100
E : 01
F : 1110
G : 1111

Expected no. of bits = 3.

6> A, F, H

7) ??? = 216

? = 4



$((A_1(A_2(A_3A_4)))A_5)$

8) LCS: 01210, length = 5.

9) LIS: 1 3 4 4 5

10) LSIS: 1 2 3 4 5

11)

i	s
5	9
2	4
1	3

12)

i	s	f	v	p	m
1	1	4	1	0	1
2	2	5	5	0	5
3	5	6	3	2	8
4	4	7	8	1	9
5	7	9	1	4	10
6	8	11	2	4	11
7	10	15	3	5	13
8	14	17	1	6	13
9	14	20	4	6	15
10	19	25	1	8	15

13) items chosen: 1, 3, 0

14) items chosen: 3, 2, 0.