Shoshi
1104107

$n \rightarrow$ no of tokens, r = rank $\rightarrow$ 1 to t. $\rightarrow$ type of words.

$n_r \rightarrow$ no of tokens for r ranks.
r type এর token কতবার আছে।

Zipf's first law: Goal is counting the number of times, n_r, particular types occure in a sampk of n tokens where r denotes rank.

Most frequently occcuring type will be assigned rank 1, second frequent type rank 2 and so on. If we assume there are t word types of n tokens then,

$$\sum_{r=1}^{t} n_r = n$$

Absolute frequenty $= n_r$ for type r

Relative frequency, $f_r = \frac{n_r}{n}$ ie

so, Zipf's 1st law, $f_r r^a = \text{constant} = c$.

'a' is slope approximated to 1.

$$f_r r = c \quad \text{---------} \quad (1)$$

or, $\frac{n_r}{n} \cdot r = c$ or, $n_r = \frac{c \cdot n}{r}$,

It can be written

$n_r = \frac{cn}{r}$, this is Zipf's first law.

2) Derivation of length equation:- From Zipf's first law.

$$n_r = \frac{C}{r}$$

$$\sum_{r=1}^{t} C_n \sum_{r=1}^{t} \frac{1}{r} \quad \text{.....(1)}$$

now, $\sum_{r=1}^{t} \frac{1}{r} = 0.5772 + \ln t + \frac{1}{2t} + \dots$

from equation (1)

$$n_r = C_n (0.5772 + \ln t)$$

$$C \simeq \frac{1}{0.5772 + \ln t}$$

consider that smallest rank, (where $r_{max} = t$) is

$$n_{r_{max}} = 1.$$

$$n_{max} = \frac{C}{r_{max}}$$

$$C = \frac{t}{n}$$

$$n = t(0.5772 + \ln t)$$

$$\text{or, } 1 = \frac{C}{t}$$

$$\text{or } C = \frac{t}{n}.$$

$$\text{or, } \frac{1}{(0.5772 + \ln t)} = \frac{t}{n}$$

* word type এবং word length কত হবে
* 100 টি different word type. (t = word type)

$$n = 100 (0.5772 + \ln 100)$$

n ↓ word length

$= 512$ words,

3) Zipf's 2nd law:

from first law, $n_r = \frac{C_n}{r}$ ----- (1)

now, $\left|\frac{dn_r}{dr}\right| = \left|\frac{-C_n}{r^2}\right| = \frac{C_n}{r^2} = \frac{C_n}{r} \cdot \frac{C_n}{r} \cdot \frac{1}{C_n} = \frac{n_r^2}{C_n}$

[Eliminating r]

---- (2)

In the tail of Zipf's law, there are several identical n_r values, thus there is a plateau of t_k types.

where $n_r = k$.

The curve becomes a staircase function and we can write,

$$\frac{dn_r}{dr} = \frac{1}{t_k}$$

In the tail we assume that,

$n_r(max) = 1$ and $k = 1$

$n_r\ max-1 = 2$ and $k = 2$

Thus in the region, $n_r = k$.

$$t_k = \frac{C_n}{k^2}.$$

4] Derivation of alternate length equation:

$$t = \sum_{k=1}^{n} \frac{c_n}{k^2} \approx \sum_{k=1}^{\infty} \frac{c_n}{k^2} = \frac{\pi^2}{6} c_n.$$

$$\therefore n = \frac{6}{\pi^2}(0.5772 + \ln t)t. \quad \left[\text{~~$c=(0.5772+\ln t)$~~} \; c=(0.5772+\ln t)\right]$$

5] Estimation of token at the begining of design:

3 steps are followings-

(1) Estimate the no of operator types.

(2) Estimate the no of operand types

(3) Summing the estimates of steps 1 and 2 to obtain the value of t and substituting in

$$n = t(0.5772 + \ln t).$$

6] Halsted Formula (software physics)

$$N = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$$

where $N \rightarrow$ program length

$\eta_1 \rightarrow$ no of operator types

$\eta_2 =$ " " operand "

from Zipfs law

$$n = t(0.5772 + \ln t)$$

Here, $t = (\eta_1 + \eta_2)(0.5772 + \ln(\eta_1 + \eta_2))$

there are three cases,

$\eta_1 \neq \eta_2$	$\eta_2 \neq \eta_1$	$\eta_1 = \eta_2 = \eta$
$N = \eta_1(0.5772 + \ln \eta_1)$	$N = \eta_2(0.5772 + \ln \eta_2)$	$N = 2\eta(0.5772 + \ln 2\eta)$

We find $\eta_1 \neq \eta_2$.

$N = \eta_1(0.5772 + \ln \eta_1)$ in Zipf's law

and ~~find~~ if $\eta_1 \neq \eta_2$

Habsted formula becomes

$N = \eta_1 \log_2 \eta_1$

Zipf's $\rightarrow \dfrac{\ln \eta_1}{\log_2 \eta_1}$ $\rightarrow \dfrac{1}{\log_2 e} = \ln 2 = 0.693 \approx 70\%$ $\rightarrow$ $\log_2 e$

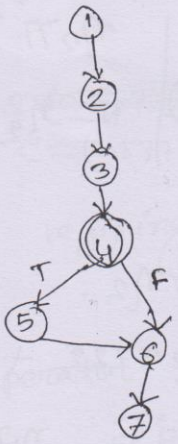
Habstead

$\therefore$ Zipf's length is 30% less than habstead formula.

Testing

(1)

1. Start
2. input a, b
3. c = a+b.
4. If (c > 100)
5. print ("it is done").
6. end if
7. stop.

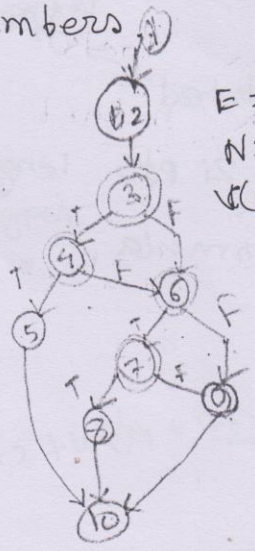


E = 7
N = 7
V(G) = 7-7+2
= 2.

Path	Test case
1. 1-2-3-4-5-6-7	a = 80, b = 30.
2. 1-2-3-4-6-7	a = 20, b = 20.

(2) Find max between 3 numbers

1. Start
2. Read a, b, c.
3. If (a > b &&
4. a > c)
5. Print (max = a).
6. else if (b > a &&
7. b > c)
8. Print ("max = b");
9. else print ("max = c");
10. ~~Retu~~ END.



E = 13
N = 10
V(G) = 12-9+2
= 5.

Path	Test case
1) 1-2-3-4-5-10	a=3, b=2, c=1
1-2-3-4-6	a=1, b=5, c=1
2) 1-2-3-4-6-7-8-10	a=3, b=2, c=4
3) 1-2-3-6-7-8-10	a=2, b=3, c=4
4) 1-2-3-4-6-7-10	a=3, b=4, c=2
5) 1-2-3-4-6-	

1-2-3-4-5-10

1-2-3-4-6-7-8-10

1-2-3-4-6-7-9-10

1-2-3-4-6-8-10

1-2-3-6-9-10

1-2-3-6-7-8-10

1-2-3-6-7-8-10

1-3-6-9-10

a=3, b=2, c=1

a=3, b=2, c=1

a=3, b=2 c=4

a=2, b=1, c=1

a=2, b=4, c=1

a=2, b=4 c=3

③

③ Read a, b, c

2. if (a ≤ 0 ||
3. b ≤ 0 ||
4. c ≤ 0))

5. type = "Bad input". go to end

6. if (a >= b+c ||
7. b >= c+a ||
8. c >= a+b)

9. type = "not a triangle". go to end

10. if (x = y ||
11. z = 2 ||
12. y = 2))

13. type = "Isosceles"
"Isosceles", go to end

14. If (x = y &&
15. y = z)

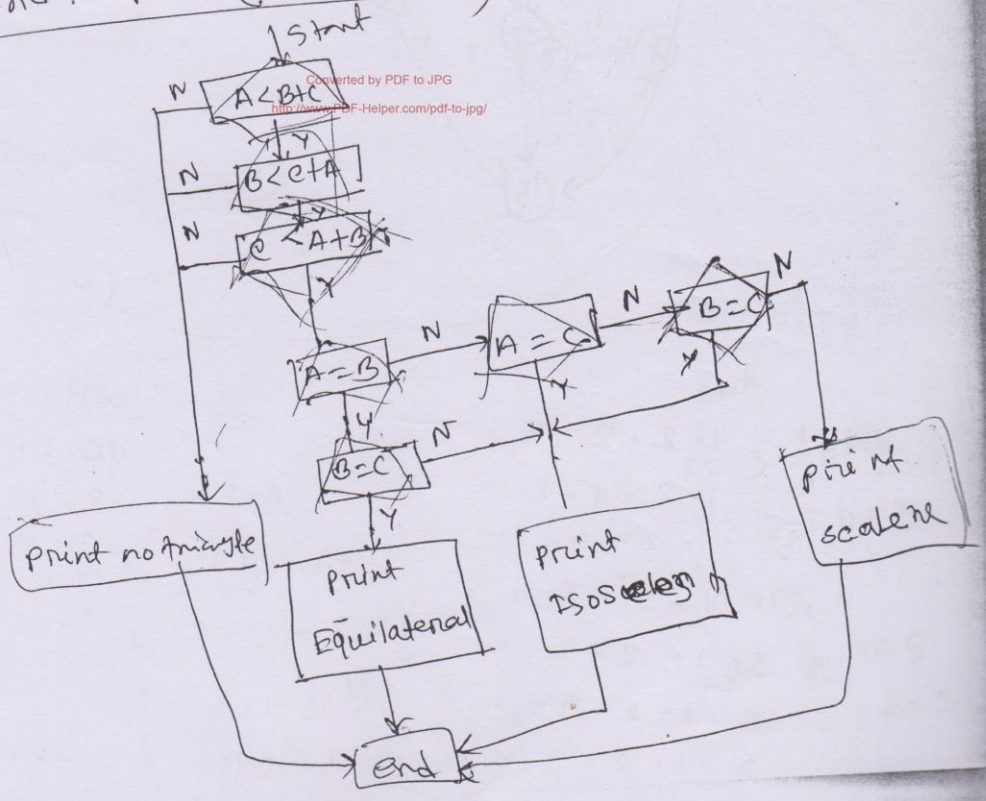
16. type = "Equilatercal". go to end

17. type = "scalene" go to end.

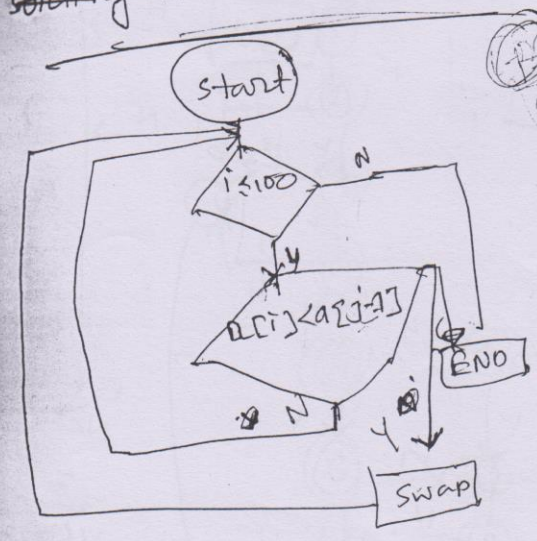
18. end: print (type).

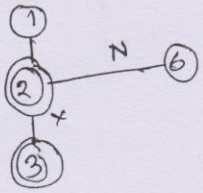
max of 3

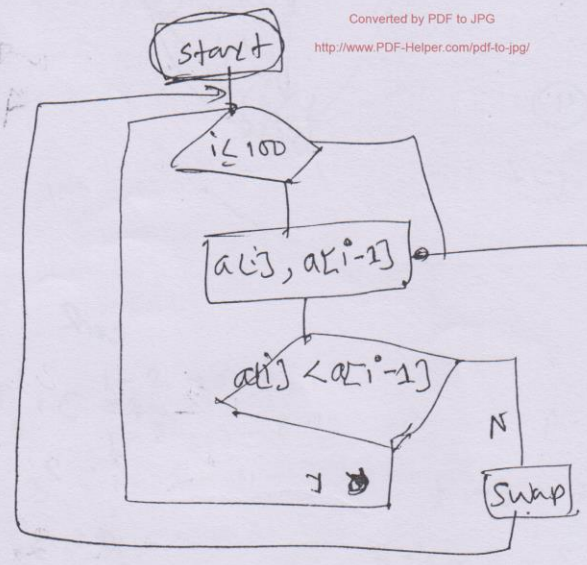
1. Read a, b, c
2. If (a > b &&
3. a > c)
4. max = a, go to end.
5. If (c > b &&
6. c > a)
7. max = c , go to end.
8. ~~else~~ else max = b.
9. end: print(max ~~= max~~)

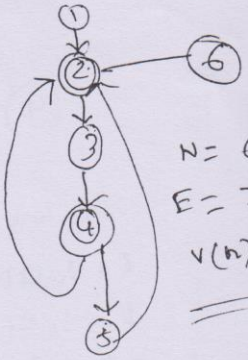


greatest of ~~sorting~~ 100 numbers.









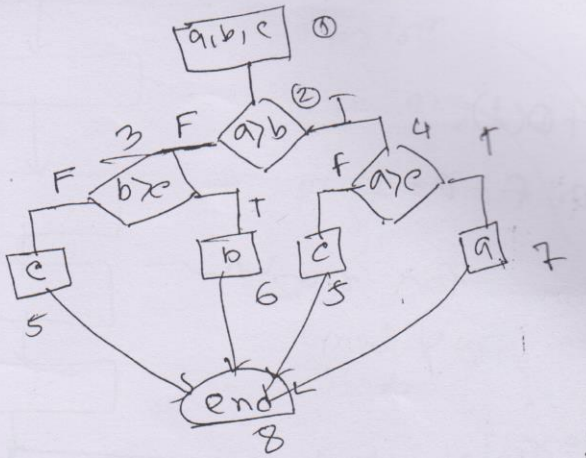
N = 6
E = 7
V(n) = 3.

path-1 1-2-6

path-2 1-2-3-4-3-6

path-3 1-2-3-4-5-2-6

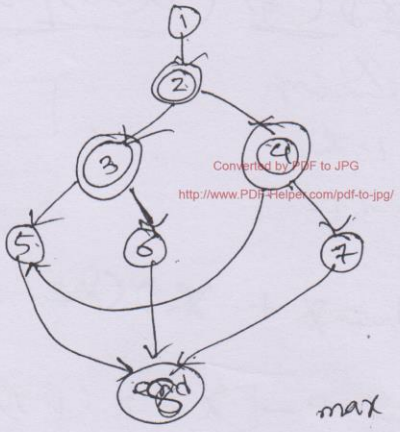
Finding greatest number



E = 10

N = 8

complexity = 10-8+2
= 4



1-2-3-5-8

1-2-3-6-8

1-2-4-5-8

1-2-4-7-8

max

c

b

a=2, b=3, c=6

a=2, b=3, c=1

a=3, b=2, c=5

a=5, b=2, c=3

$f = m + D$

$f = m(t) + D(t)$.

$t = 0$, $m = 0$, $f = D = 1$.

x = fraction for mortar

$x = m/f$.

$m(t_1) = x\, D(t_1)$ $\quad (D(t_1) = 1)$

for 1st project. $= x$.

$D = 1 - x$

2nd $\quad m(t_2) = x + x\, D(t_2) = x + x(1-x)$

$D = 1 - [x + x(1-x)]$

$= 1 - x - x + x^2$

$= 1 - 2x + x^2$

$= (1-x)^2$

and so on;

for kth proj $\quad D = (1-x)^k$

$m = 1 - D = 1 - (1-x)^k$