Sqrt of Num

I/p → Number → x

o/p → $\sqrt{x}$ → ans

I/p → 25 36 50

$\downarrow$ $\downarrow$ $\downarrow$

$\sqrt{25}$ $\sqrt{36}$ $\sqrt{50}$

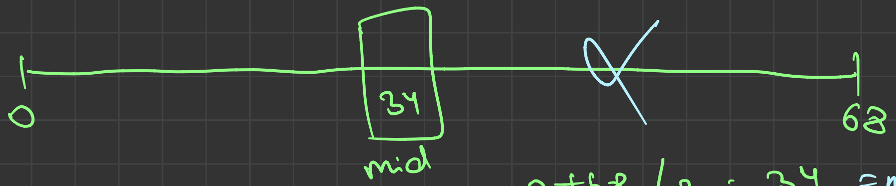
$\downarrow$ $\downarrow$ $\downarrow$

o/p → 5 6 7 → nearby

for Ex → I/p → 68

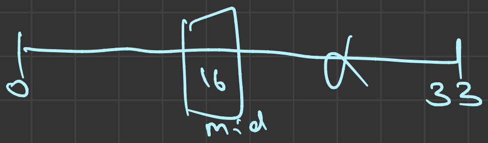
I create my own search space

from 0 → 68



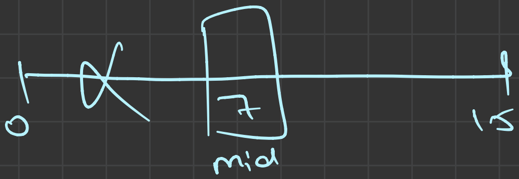
0+68 / 2 = 34 = mid

34 * 34 <= 68 ✗



0+33/2 = 16

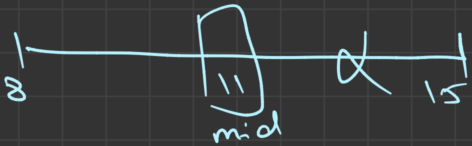
16 * 16 <= 68 ✗



$0+15/2 = 7$

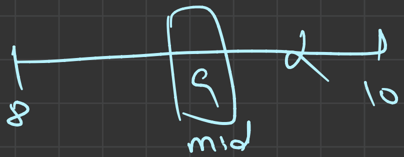
$7*7 <= 68$ ✓

ans = 7

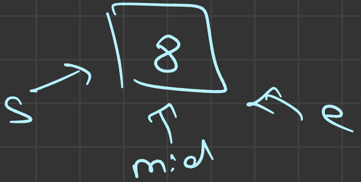


$8+15/2 = 11$

$11*11 \leq 68$ ✗



$9*9 \leq 68$ ✗



$8*8 \leq 68$ ✓

ans = 7

return ans = 8

Solve using Precision

← Using looping

$\sqrt{68}$ => $\boxed{8}$.246

$\sqrt{\ }$ we already find this

0.1

$8 \xrightarrow{add} .1 \Rightarrow (8.1)^2 \le 68$ true/save

=> $8 \longrightarrow .2 \Rightarrow (8.2)^2 \le 68$ true/save

$8 \longrightarrow .3 \Rightarrow (8.3)^2 \le 68$ false

we find 8.2 Now move further

0.01

$8.2 \longrightarrow .01 \Rightarrow (8.21)^2 \le 68$

$\vdots$

=> $8.2 \rightarrow .04 \Rightarrow (8.24)^2 \le 68$

Move further

0.001

$8.24 \rightarrow .001 \Rightarrow (8.241)^2 \le 68$

$\vdots$

=> $8.24 \rightarrow .006 \Rightarrow (8.246)^2 \le 68$

we find 8.246

```
double  morePrecision (int n, int Precision , int temp)
{
        double factor = 1 ;
        double ans = temp;

       for (int i = 0 ; i < precision ; i++)
       {
            factor = factor / 10 ;

          for( double j = ans ; j*j < n ; j+=factor)
          {
               ans = j ;
          }
       }
}


morePrecision ( 68 , 3 , 8 ) ;
```

```
#include <bits/stdc++.h>
int sqrt(long long n){
    int s = 0;
    int e = n/2;

    int m = s + (e-s)/2;

    int ans = -1;

    while(s <= e){
        if((m*m) <= n){
            ans = m;
            s = m+1;
        }
        else{
            e = m-1;
        }
        m = s + (e-s)/2;
    }
    return ans;
}

double morePrecision(int n, int precision, int temp){
    double factor = 1;
    double ans = temp;

    for(int i = 0; i < precision; i++){
        factor = factor/10;
        for(double j = ans; j*j <= n; j+=factor){
            ans = j;
        }
    }
    return ans;
}



double squareRoot(long long n, int d) {
    int temp = sqrt(n);
    double ans = morePrecision(n,d,temp);
    return ans;
}
```