Square root of an integer

Given an integer x, find square root of it. If x is not a perfect square, then return floor(√x).

Examples:

```
Input: x = 4
Output: 2

Input: x = 11
Output: 3
```

We strongly recommend you to minimize your browser and try this yourself first.

There can be many ways to solve this problem. For example Babylonian Method is one way.

A **Simple Solution** to find floor of square root is to try all numbers starting from 1. For every tried number i, if i*i is smaller than x, then increment i. We stop when i*i becomes more than or equal to x. Below is C++ implementation of above idea.

```
// A C++ program to find floor(sqrt(x)
#include<bits/stdc++.h>
using namespace std;

// Returns floor of square root of x
int floorSqrt(int x)
{
    // Base cases
    if (x == 0 || x == 1)
       return x;

    // Staring from 1, try all numbers until
    // i*i is greater than or equal to x.
    int i = 1, result = 1;
    while (result < x)
    {
       if (result == x)
          return result;
       i++;
       result = i*i;
    }
    return i-1;
}

// Driver program
int main()
{
    int x = 11;
    cout << floorSqrt(x) << endl;
    return 0;
}
```

Run on IDE

Output:

```
3
```

Time complexity of the above solution is O(√ n). Thanks Fattepur Mahesh for suggesting this solution.

A **Better Solution** to do Binary Search.

```
Let  's' be the answer.  We know that 0 <=  s <= x.

Consider any random number r.

    If r*r <= x, s >= r

    If r*r > x, s < r.
```

Algorithm:

1) Start with 'start' = 0, end = 'x',

2) Do following while 'start' is smaller than or equal to 'end'.

a) Compute 'mid' as (start + end)/2

b) compare mid*mid with x.

c) If x is equal to mid*mid, return mid.

d) If x is greater, do binary search between mid+1 and end. In this case, we also update ans (Note that we need floor).

e) If x is smaller, do binary search between start and mid-1

Below is C++ implementation of above idea.

C/C++ | Java

```
// A Java program to find floor(sqrt(x)
public class Test
{
    public static int floorSqrt(int x)
    {
        // Base Cases
        if (x == 0 || x == 1)
            return x;

        // Do Binary Search for floor(sqrt(x))
        int start = 1, end = x, ans=0;
        while (start <= end)
        {
            int mid = (start + end) / 2;

            // If x is a perfect square
            if (mid*mid == x)
                return mid;

            // Since we need floor, we update answer when mid*mid is
            // smaller than x, and move closer to sqrt(x)
            if (mid*mid < x)
            {
                start = mid + 1;
                ans = mid;
            }
            else   // If mid*mid is greater than x
                end = mid - 1;
        }
        return ans;
    }

    // Driver Method
    public static void main(String args[])
    {
        int x = 11;
        System.out.println(floorSqrt(x));
    }
}
// Contributed by InnerPeace
```

Run on IDE

Output:

```
3
```

Time Complexity: O(Log x)

Thanks to Gaurav Ahirwar for suggesting above method.

Note: The Binary Search can be further optimized to start with 'start' = 0 and 'end' = x/2. Floor of square root of x cannot be more than x/2 when x > 1.