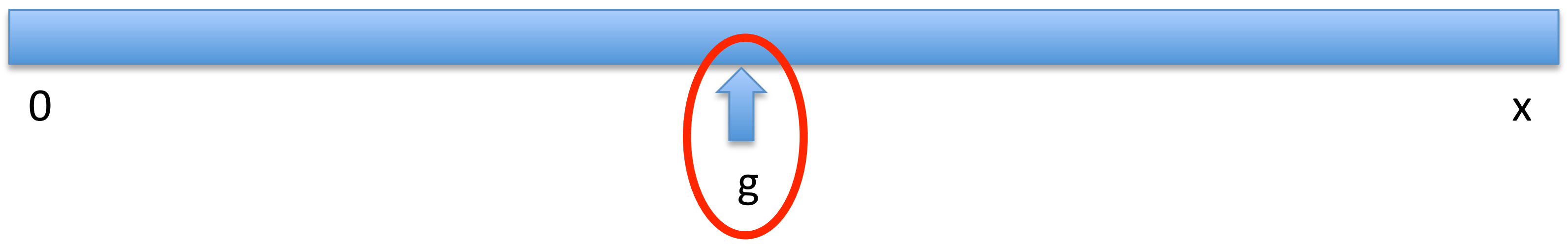
Bisection search

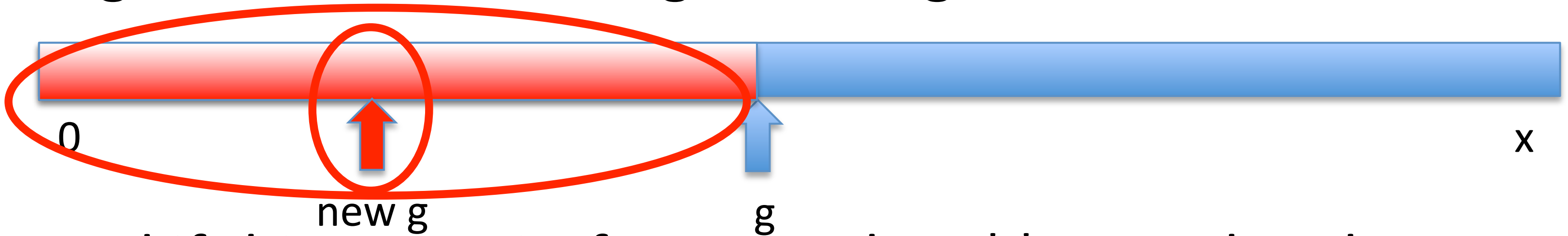
- We know that the square root of x lies between 0 and x, from mathematics
- Rather than exhaustively trying things starting at 0, suppose instead we pick a number in the middle of this range



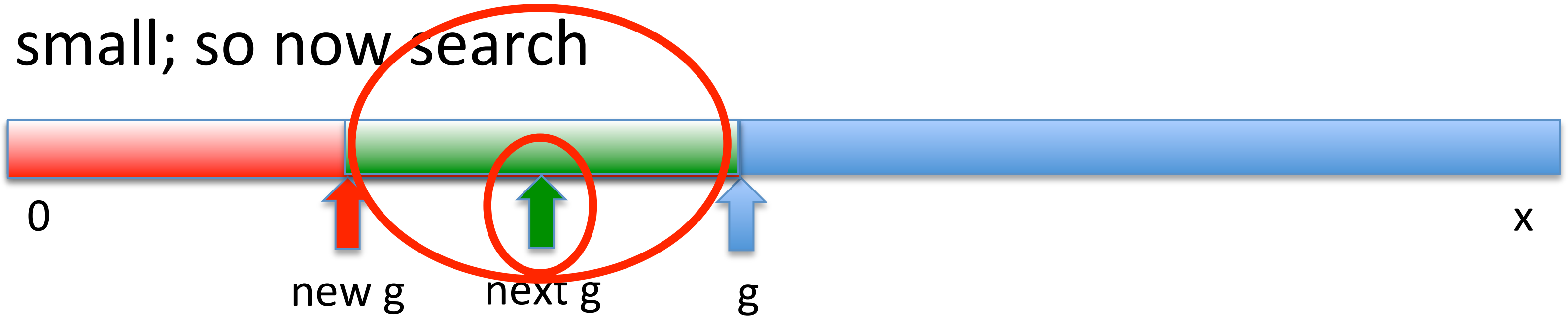
- If we are lucky, this answer is close enough

Bisection search

- If not close enough, is guess too big or too small?
- If g**2 > x, then know g is too big; but now search



- And if this new g is, for example, g**2 < x, then know too small; so now search



- At each stage, reduce range of values to search by half

Example of square root

Too far apart

```
x = 25
epsilon = 0.01
numGuesses = 0
low = 0.0
high = x
ans = (high + low)/2.0
while abs(ans**2 - x) >= epsilon:
    print('low = ' + str(low) + ' high = ' + str(high) + ' ans = ' + str(ans))
    numGuesses += 1
    if ans**2 < x:
        low = ans
    else:
        high = ans
    ans = (high + low)/2.0
print('numGuesses = ' + str(numGuesses))
print(str(ans) + ' is close to square root of ' + str(x))
```

Some observations

- Bisection search radically reduces computation time – being smart about generating guesses is important
- Should work well on problems with "ordering" property – value of function being solved varies monotonically with input value
  - Here function is g**2; which grows as g grows