Leetcode: Data Structure and Algorithms

Rafiq Islam

Table of contents

Binary Search

Problem 1: Leetcode 69: Sqrt(x)

Given a non-negative integer, x, return the square root of x rounded down to the nearest integer. The returned integer should be non-negative as well.
You may not use any built-in exponent function. For example, `x**0.5` in python.
Example:

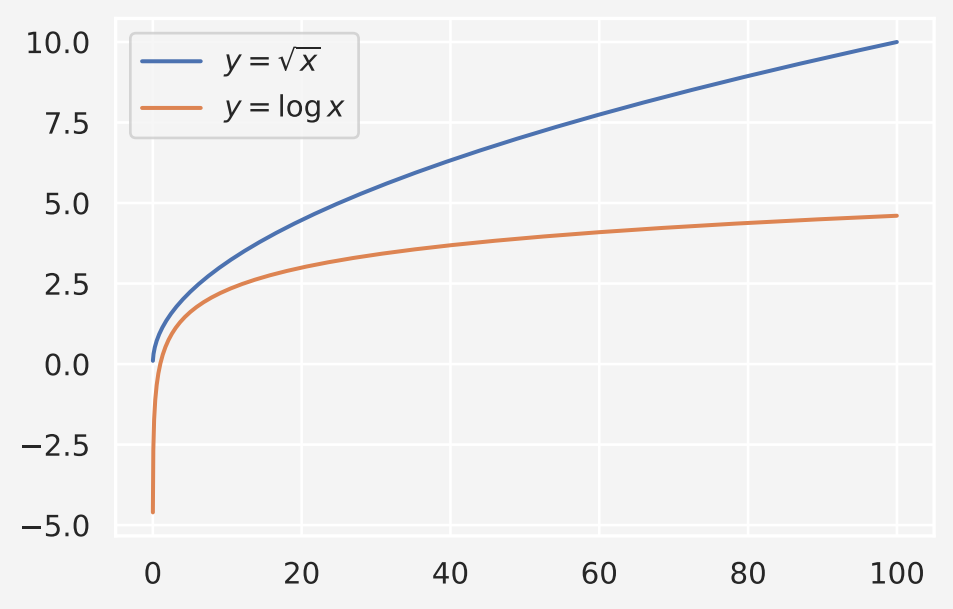
```
Input: x=4
Output: 2

Input: x=8
Output: 2
```

Explanation: Square root of 4 is 2 and square root of 8 is 2.8284. But we need to round down to any fraction. Therefore, the square root of 8 is also 2.
Solution:
The square root of any number $x \geq 0$ is less than or equal to x. The brute force solution to this would be $\mathcal{O}(\sqrt{n})$. Because, say $x = 8$, then

for $i = 1$ to 8:

$$1^2 = 1 < 8$$
$$2^2 = 4 < 8$$
$$3^2 = 9 > 8$$



In contrast, if we explore binary search then the time complexity reduces to $\mathcal{O}(\log n)$. Say the square root is s which is the middle value in the range of 1 to x. Then if $s^2 > x$, we search for the root in the left half. Otherwise, if $s^2 < x$ then we search the right side. However, when $s^2 < x$, then s is a possible candidate for the square root.
Algorithm:

1. set left value $l = 0$, right value $r = x$
2. Compute the middle value $m = l + (r - l)/2$
3. If $m^2 > x$ then search the left side: set $r = m - 1$
4. If $m^2 < x$ then search the right side: set $l = m + 1$

```
def square_root(x):
  l, r = 0, x
  sq = 0
  while l<=r:
    m = l + (r-l)//2
    if m**2 > x:
      r= m-1
    elif m**2 < x:
      l = m+1
      sq = m
    else:
      return m
  return sq

print(square_root(6))
```

2

Array

Problem 1: Intersection of two sets

Say, we are given two sets $A = [2, 3, 5, 6, 8]$ and $B = [4, 6, 8]$. We want to find the intersection of the elements in these sets.

```
def intersection_of_two_sets(A,B):
  set_a = set(A)
  return [b for b in B if b in set_a]

A = [2,3,5,6,8]
B = [4,6,8]
print(intersection_of_two_sets(A,B))
```

[6, 8]

Problem 2: Histogram from a given array and bin number

Say, we are given an array $A = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]$ and number of bins. We want to

```
def generate_histogram(A, num_bins):
  min_value = min(A)
  max_value = max(A)
  bin_width = (max_value-min_value)/num_bins
```

String

Problem 1: Leetcode 242: Valid Anagram

Given two strings `s` and `t`, return `true` if `t` is an *anagram* of `s`, and `false` *otherwise*
Example: Input: `s="anagram"`, `t="nagaram"` Output: `true`
Since the word `anagram` has 3 a's, 1 n, 1 g, 1 r, and 1 m and `nagaram` has exactly the same number of the same alphabets, therefore they are anagram of each other.
Example: Input: `s="rat"`, `t="cat"` Output: `false`
Since the word `rat` has 1 r, 1 a, and 1 t but `cat` has 2 elements same as `rat` but one element different. Therefore the answer is false.
Note that, they both have the same length.
Constraints:

- $1 \leq s.\text{length}, t.\text{length} \leq 5 \times 10^4$
- `s` and `t` consists of lowercase English letters

Solution:
We can use `hasmap` to solve this problem. Basically, we will create two hasmaps for two words and match the keys and values of the hasmaps. If they are equal then it's an anagram, otherwise not.

```python
def isAnagram(s,t):
  if len(s) != len(t):
    return False

  hash_s, hash_t = {}, {}
  for i in range(len(s)):
    hash_s[s[i]] = 1 + hash_s.get(s[i],0)  # get function collects the key and values.
    hash_t[t[i]] = 1 + hash_t.get(t[i],0)  # if there's no key, 0 is the default value

  for c in hash_s:
    if hash_s[c] != hash_t.get(c,0):       # Here, get function ensures there is no
      return False                         # key error
  return True

s = "anagram"
t = "nagaram"

print(isAnagram(s,t))

u = "rat"

print(isAnagram(s,u))
```

```
True
False
```

Time and Space Complexity:
Time complexity $\mathcal{O}(m+n)$ where m and n are the length of `s` and `t` and memory complexity is the same $\mathcal{O}(m+n)$
Optimization: Can we solve the problem in $\mathcal{O}(1)$? If we assume sort doesn't require extra space, then

```python
def isAnagram(s,t):
  return sorted(s)== sorted(t)

s= "anagram"
t= "nagaram"

print(isAnagram(s,t))
```

```
True
```

Reference

All my solutions here are based on the solutions found from NeetCode.