Assignment – 12/06

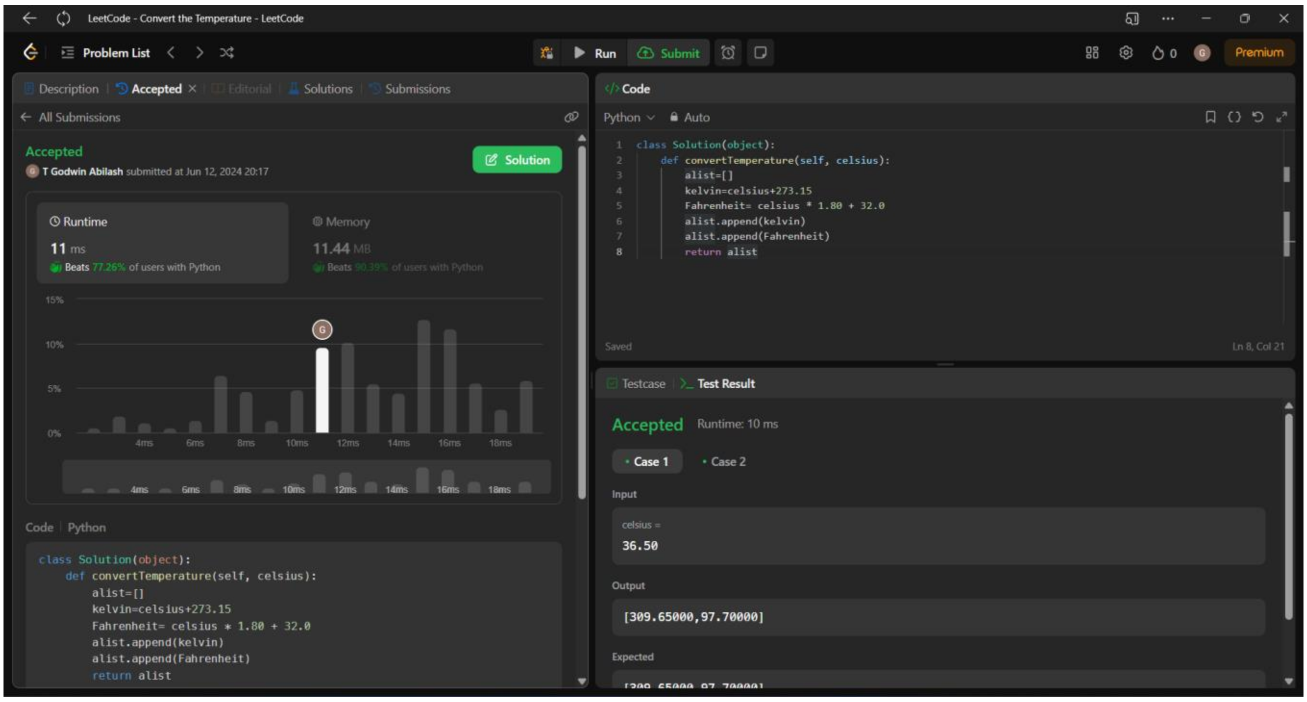
T Godwin Abilash – 192321012

1. Convert the Temperature

Code:

```
class Solution(object):
    def convertTemperature(self, celsius):
        alist=[]
        kelvin=celsius+273.15
        Fahrenheit= celsius * 1.80 + 32.0
        alist.append(kelvin)
        alist.append(Fahrenheit)
        return alist
```

Screenshot:



Accepted
T Godwin Abilash submitted at Jun 12, 2024 20:17

Runtime
11 ms
Beats 77.26% of users with Python

Memory
11.44 MB
Beats 90.39% of users with Python

Code | Python

```
class Solution(object):
    def convertTemperature(self, celsius):
        alist=[]
        kelvin=celsius+273.15
        Fahrenheit= celsius * 1.80 + 32.0
        alist.append(kelvin)
        alist.append(Fahrenheit)
        return alist
```

```
class Solution(object):
    def convertTemperature(self, celsius):
        alist=[]
        kelvin=celsius+273.15
        Fahrenheit= celsius * 1.80 + 32.0
        alist.append(kelvin)
        alist.append(Fahrenheit)
        return alist
```

Accepted Runtime: 10 ms

Input

```
celsius =
36.50
```

Output

```
[309.65000,97.70000]
```

Expected

```
[309.65000,97.70000]
```

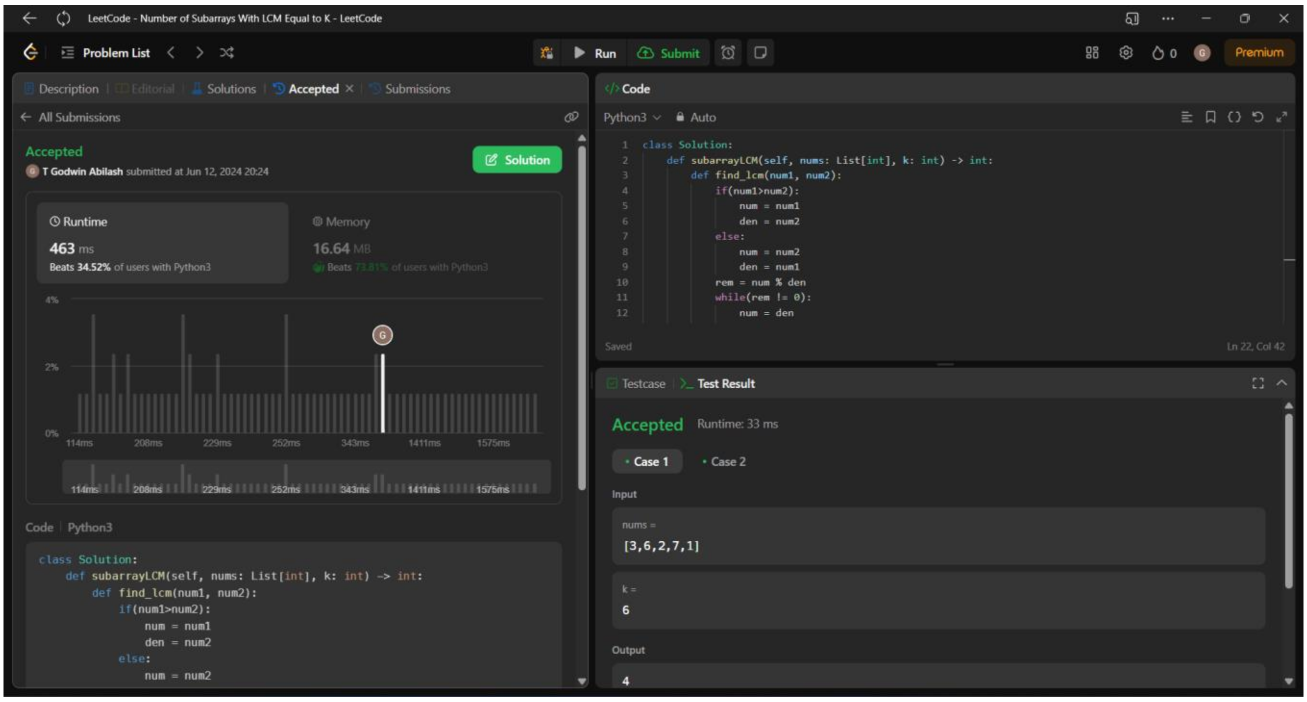
Time Complexity: O(n)

2. Number of Subarrays With LCM Equal to K

Code:

```
class Solution:
    def subarrayLCM(self, nums: List[int], k: int) -> int:
        def find_lcm(num1, num2):
            if(num1>num2):
                num = num1
                den = num2
            else:
                num = num2
                den = num1
            rem = num % den
            while(rem != 0):
                num = den
                den = rem
                rem = num % den
            gcd = den
            lcm = int(int(num1 * num2)/int(gcd))
            return lcm
        count=0
        for i in range(len(nums)):
            lcm=1
            for j in range(i,len(nums)):
                lcm=find_lcm(nums[j],lcm)
                if lcm==k:
                    count+=1
                if lcm>k:
                    break
        return count
```

Screenshot for I/O:


LeetCode - Number of Subarrays With LCM Equal to K - LeetCode
Problem List
Run
Submit
Premium
Description | Editorial | Solutions | Accepted | Submissions
All Submissions
Solution
Code
Python3
Auto
Saved
Ln 22, Col 42
Testcase
Test Result

Accepted
T Godwin Abilash submitted at Jun 12, 2024 20:24

Runtime
463 ms
Beats 34.52% of users with Python3

Memory
16.64 MB
Beats 73.81% of users with Python3

Code | Python3

```
class Solution:
    def subarrayLCM(self, nums: List[int], k: int) -> int:
        def find_lcm(num1, num2):
            if(num1>num2):
                num = num1
                den = num2
            else:
                num = num2
                den = num1
            rem = num % den
            while(rem != 0):
                num = den
```

Accepted Runtime: 33 ms

Case 1 · Case 2

Input

```
nums =
[3,6,2,7,1]
k =
6
```

Output

```
4
```

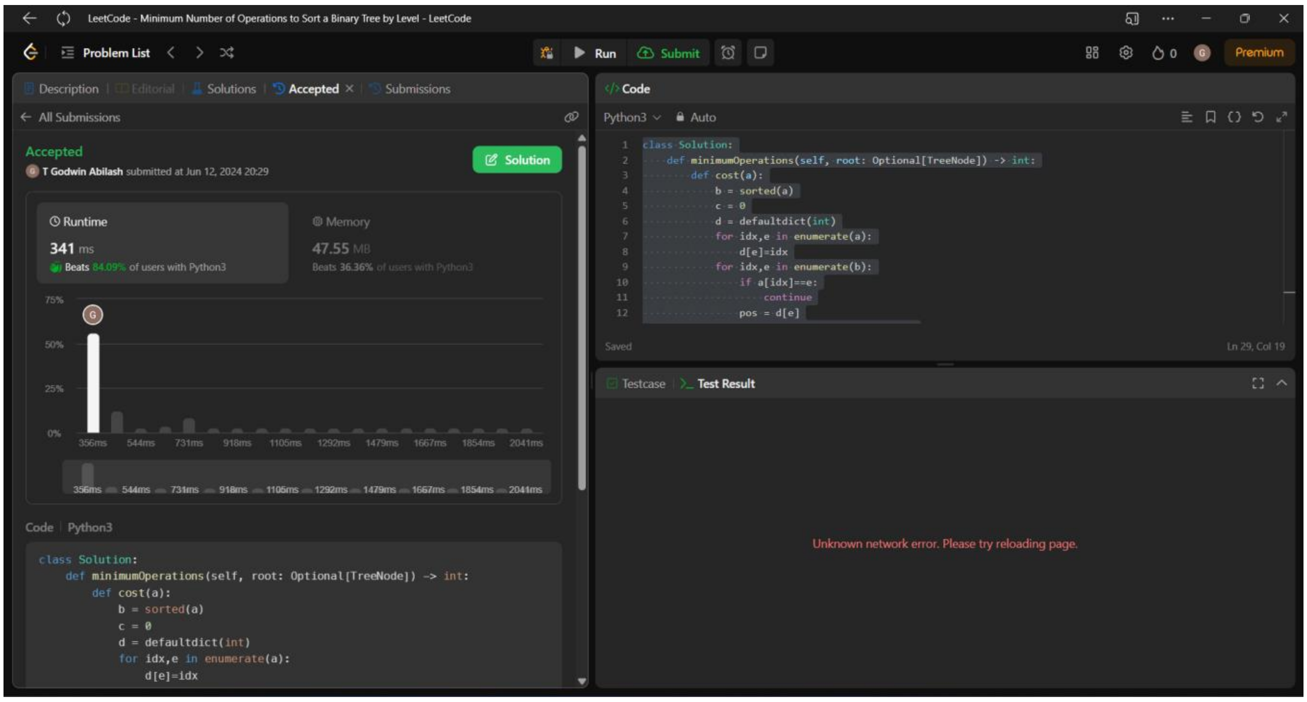
Time Complexity: O(n2)

3. Minimum Number of Operations to Sort a Binary Tree by Level

Code:

```
class Solution:
    def minimumOperations(self, root: Optional[TreeNode]) -> int:
        def cost(a):
            b = sorted(a)
            c = 0
            d = defaultdict(int)
            for idx,e in enumerate(a):
                d[e]=idx
            for idx,e in enumerate(b):
                if a[idx]==e:
                    continue
                pos = d[e]
                a[idx],a[pos] = a[pos],a[idx]
                d[a[pos]] = pos
                c+=1
            return c
        d = defaultdict(list)
        dfs = [(root,1)]
        while dfs:
            node,h = dfs.pop()
            d[h].append(node.val)
            if node.right:
                dfs.append((node.right,h+1))
            if node.left:
                dfs.append((node.left,h+1))
        ans = 0
        for m in d.values():
            ans += cost(m)
        return ans
```

Screenshot:



Accepted
T Godwin Abilash submitted at Jun 12, 2024 20:29

Runtime
341 ms
Beats 84.09% of users with Python3

Memory
47.55 MB
Beats 36.36% of users with Python3

Code | Python3

```
class Solution:
    def minimumOperations(self, root: Optional[TreeNode]) -> int:
        def cost(a):
            b = sorted(a)
            c = 0
            d = defaultdict(int)
            for idx,e in enumerate(a):
                d[e]=idx
```

```
class Solution:
    def minimumOperations(self, root: Optional[TreeNode]) -> int:
        def cost(a):
            b = sorted(a)
            c = 0
            d = defaultdict(int)
            for idx,e in enumerate(a):
                d[e]=idx
            for idx,e in enumerate(b):
                if a[idx]==e:
                    continue
                pos = d[e]
```

Saved
Ln 29, Col 19

Unknown network error. Please try reloading page.

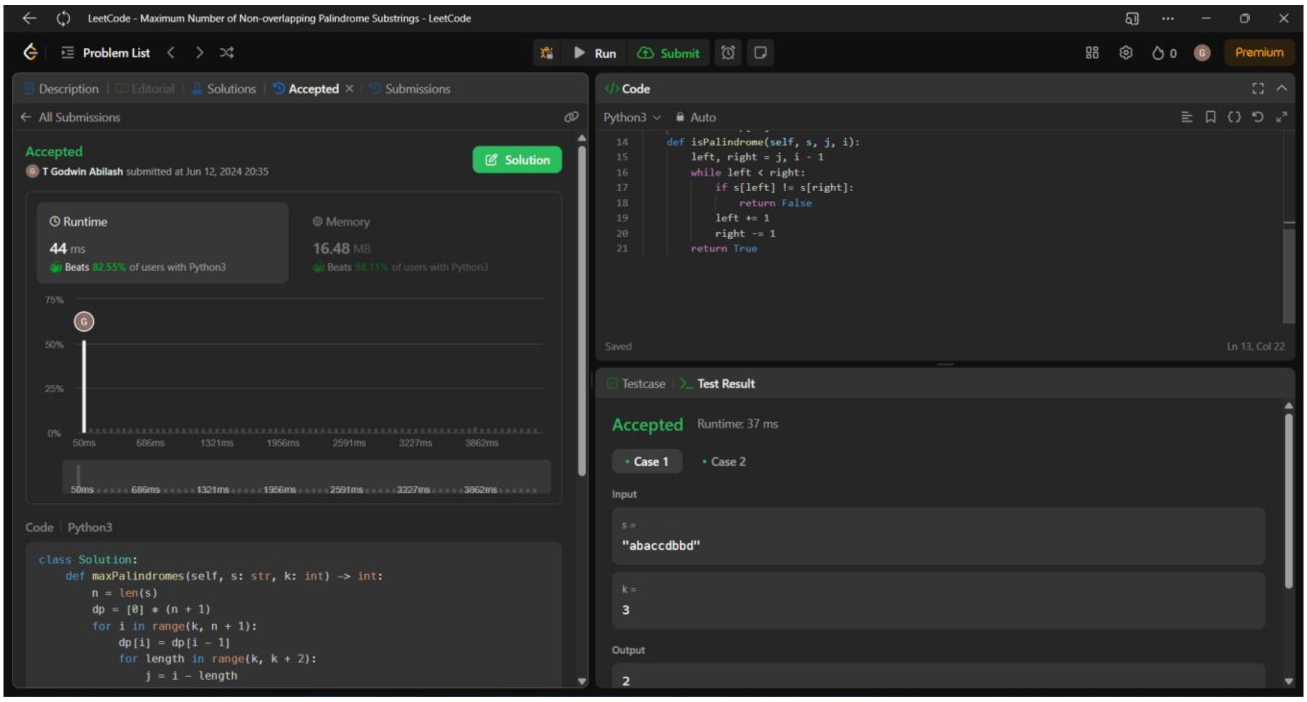
Time Complexity: O(n2)

4. Maximum Number of Non-overlapping Palindrome Substrings

Code:

```
class Solution:
    def maxPalindromes(self, s: str, k: int) -> int:
        n = len(s)
        dp = [0] * (n + 1)
        for i in range(k, n + 1):
            dp[i] = dp[i - 1]
            for length in range(k, k + 2):
                j = i - length
                if j < 0:
                    break
                if self.isPalindrome(s, j, i):
                    dp[i] = max(dp[i], 1 + dp[j])
        return dp[-1]
    def isPalindrome(self, s, j, i):
        left, right = j, i - 1
        while left < right:
            if s[left] != s[right]:
                return False
            left += 1
            right -= 1
        return True
```

Screenshot:


LeetCode - Maximum Number of Non-overlapping Palindrome Substrings - LeetCode

Accepted
T Godwin Abilash submitted at Jun 12, 2024 20:35

Runtime
44 ms
Beats 82.55% of users with Python3

Memory
16.48 MB
Beats 98.11% of users with Python3

Code | Python3

```
class Solution:
    def maxPalindromes(self, s: str, k: int) -> int:
        n = len(s)
        dp = [0] * (n + 1)
        for i in range(k, n + 1):
            dp[i] = dp[i - 1]
            for length in range(k, k + 2):
                j = i - length
```

```
    def isPalindrome(self, s, j, i):
        left, right = j, i - 1
        while left < right:
            if s[left] != s[right]:
                return False
            left += 1
            right -= 1
        return True
```

Saved   Ln 13, Col 22

Accepted   Runtime: 37 ms

Case 1   Case 2

Input

s =
"abaccdbbd"

k =
3

Output

2

Time Complexity: O(n)

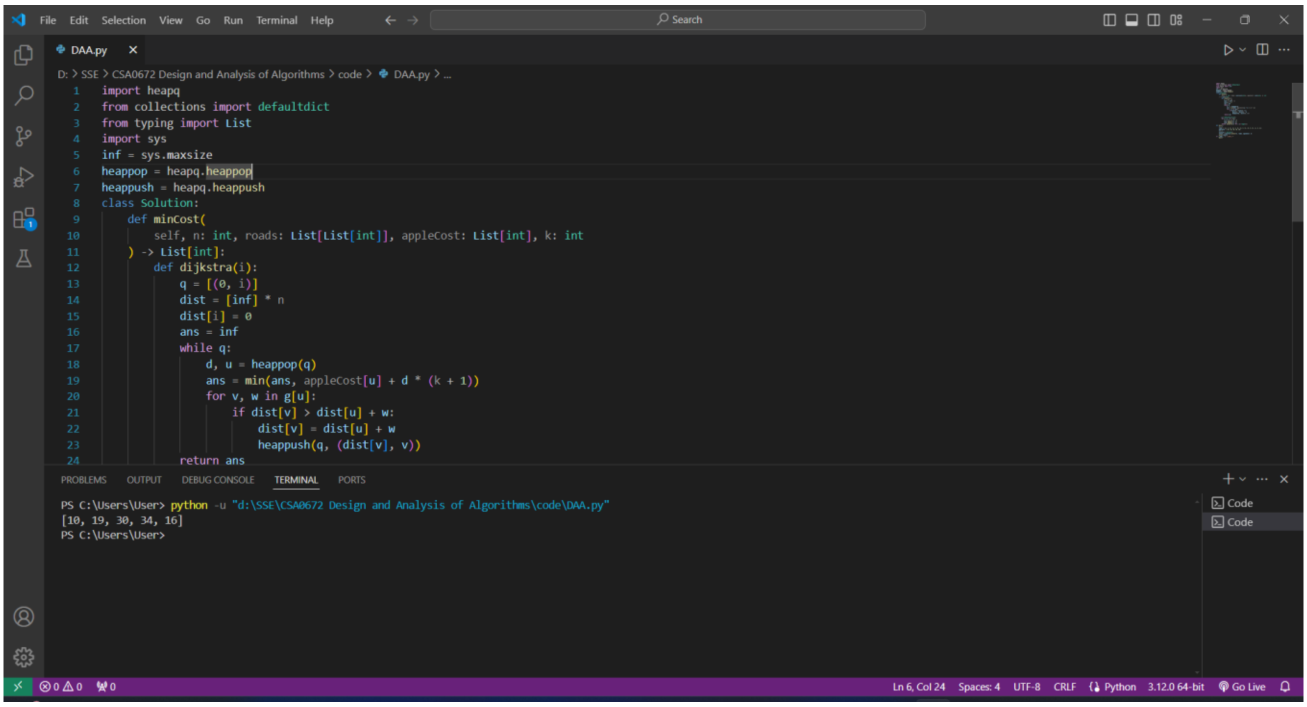
5. Minimum Cost to Buy Apples

Code:

```
import heapq
from collections import defaultdict
from typing import List
import sys
inf = sys.maxsize
heappop = heapq.heappop
heappush = heapq.heappush
class Solution:
    def minCost(
        self, n: int, roads: List[List[int]], appleCost: List[int], k:
int
    ) -> List[int]:
        def dijkstra(i):
            q = [(0, i)]
            dist = [inf] * n
            dist[i] = 0
            ans = inf
            while q:
                d, u = heappop(q)
                ans = min(ans, appleCost[u] + d * (k + 1))
                for v, w in g[u]:
                    if dist[v] > dist[u] + w:
                        dist[v] = dist[u] + w
                        heappush(q, (dist[v], v))
            return ans

        g = defaultdict(list)
        for a, b, c in roads:
            a, b = a - 1, b - 1
            g[a].append((b, c))
            g[b].append((a, c))
        return [dijkstra(i) for i in range(n)]
def main():
    n = 5
    roads = [[1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 6], [1, 5, 2]]
    appleCost = [10, 20, 30, 40, 50]
    k = 2
    solution = Solution()
    result = solution.minCost(n, roads, appleCost, k)
    print(result)
if __name__ == "__main__":
    main()
```

Screenshot:



D: > SSE > CSA0672 Design and Analysis of Algorithms > code > DAA.py > ...

```python
import heapq
from collections import defaultdict
from typing import List
import sys
inf = sys.maxsize
heappop = heapq.heappop
heappush = heapq.heappush
class Solution:
    def minCost(
        self, n: int, roads: List[List[int]], appleCost: List[int], k: int
    ) -> List[int]:
        def dijkstra(i):
            q = [(0, i)]
            dist = [inf] * n
            dist[i] = 0
            ans = inf
            while q:
                d, u = heappop(q)
                ans = min(ans, appleCost[u] + d * (k + 1))
                for v, w in g[u]:
                    if dist[v] > dist[u] + w:
                        dist[v] = dist[u] + w
                        heappush(q, (dist[v], v))
            return ans
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"
[10, 19, 30, 34, 16]
PS C:\Users\User>
```

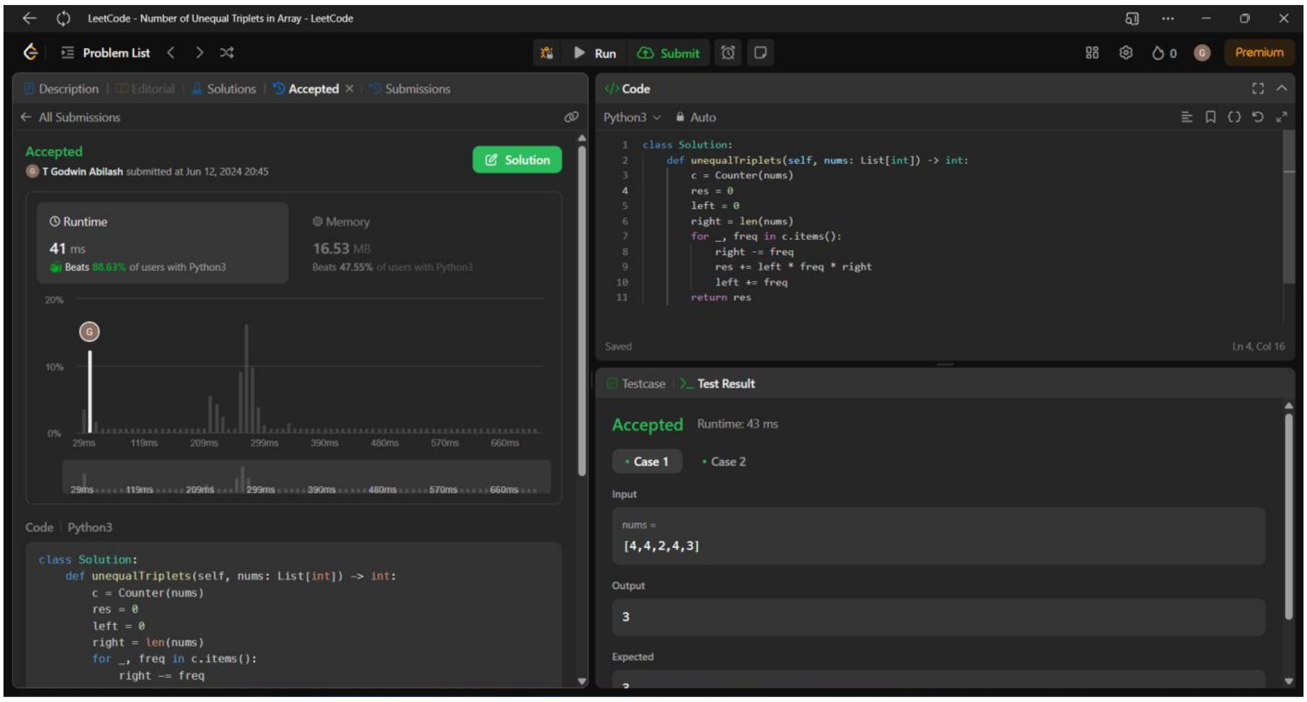
Time Complexity: O(n2)

7. Number of Unequal Triplets in Array

Code:

```
class Solution:
    def unequalTriplets(self, nums: List[int]) -> int:
        c = Counter(nums)
        res = 0
        left = 0
        right = len(nums)
        for _, freq in c.items():
            right -= freq
            res += left * freq * right
            left += freq
        return res
```

Screenshot:


LeetCode - Number of Unequal Triplets in Array - LeetCode

Accepted
T Godwin Abilash submitted at Jun 12, 2024 20:45

Runtime
41 ms
Beats 88.63% of users with Python3

Memory
16.53 MB
Beats 47.55% of users with Python3

Code | Python3

```
class Solution:
    def unequalTriplets(self, nums: List[int]) -> int:
        c = Counter(nums)
        res = 0
        left = 0
        right = len(nums)
        for _, freq in c.items():
            right -= freq
```

Code
Python3 Auto

```
class Solution:
    def unequalTriplets(self, nums: List[int]) -> int:
        c = Counter(nums)
        res = 0
        left = 0
        right = len(nums)
        for _, freq in c.items():
            right -= freq
            res += left * freq * right
            left += freq
        return res
```

Saved Ln 4, Col 16

Testcase | Test Result

Accepted Runtime: 43 ms

Case 1 Case 2

Input

```
nums =
[4,4,2,4,3]
```

Output

```
3
```

Expected

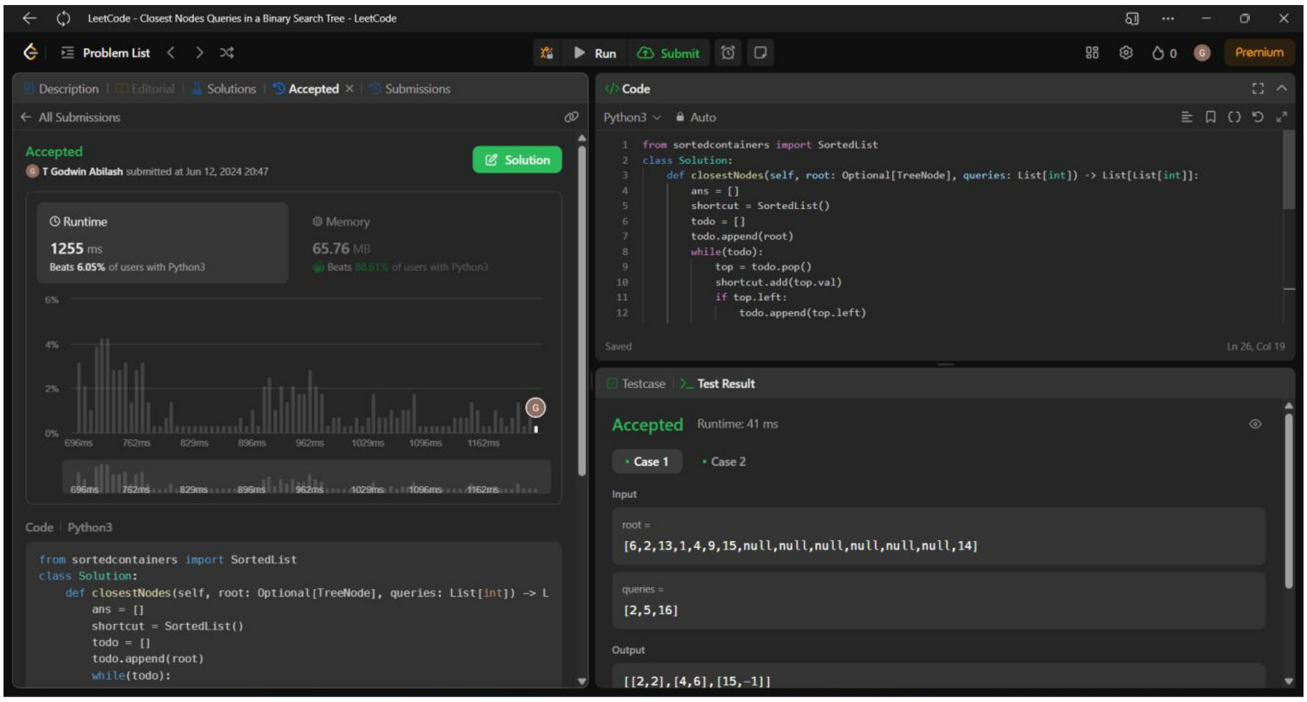
Time Complexity: O(n)

8. Closest Nodes Queries in a Binary Search Tree

Code:

```
from sortedcontainers import SortedList
class Solution:
    def closestNodes(self, root: Optional[TreeNode], queries: List[int]) -> 
List[List[int]]:
        ans = []
        shortcut = SortedList()
        todo = []
        todo.append(root)
        while(todo):
            top = todo.pop()
            shortcut.add(top.val)
            if top.left:
                todo.append(top.left)
            if top.right:
                todo.append(top.right)
        for i in range(len(queries)):
            if queries[i] in shortcut:
                ans.append([queries[i], queries[i]])
                continue
            idx = shortcut.bisect_left(queries[i])
            if idx == 0:
                ans.append([-1, shortcut[0]])
            elif idx == len(shortcut):
                ans.append([shortcut[-1],-1])
            else:
                ans.append([shortcut[idx-1], shortcut[idx]])
        return ans
```

Screenshot for I/O:


LeetCode - Closest Nodes Queries in a Binary Search Tree - LeetCode
Problem List
Run
Submit
Premium
Description | Editorial | Solutions | Accepted | Submissions
All Submissions
Solution
Code
Python3
Auto
Testcase
Test Result
Case 1
Case 2

Accepted
T Godwin Abilash submitted at Jun 12, 2024 20:47

Runtime
1255 ms
Beats 6.05% of users with Python3

Memory
65.76 MB
Beats 88.61% of users with Python3

Code | Python3

```
from sortedcontainers import SortedList
class Solution:
    def closestNodes(self, root: Optional[TreeNode], queries: List[int]) -> L
        ans = []
        shortcut = SortedList()
        todo = []
        todo.append(root)
        while(todo):
```

```
from sortedcontainers import SortedList
class Solution:
    def closestNodes(self, root: Optional[TreeNode], queries: List[int]) -> List[List[int]]:
        ans = []
        shortcut = SortedList()
        todo = []
        todo.append(root)
        while(todo):
            top = todo.pop()
            shortcut.add(top.val)
            if top.left:
                todo.append(top.left)
```

Saved   Ln 26, Col 19

Accepted   Runtime: 41 ms

Input

root =
[6,2,13,1,4,9,15,null,null,null,null,null,null,14]

queries =
[2,5,16]

Output

[[2,2],[4,6],[15,-1]]

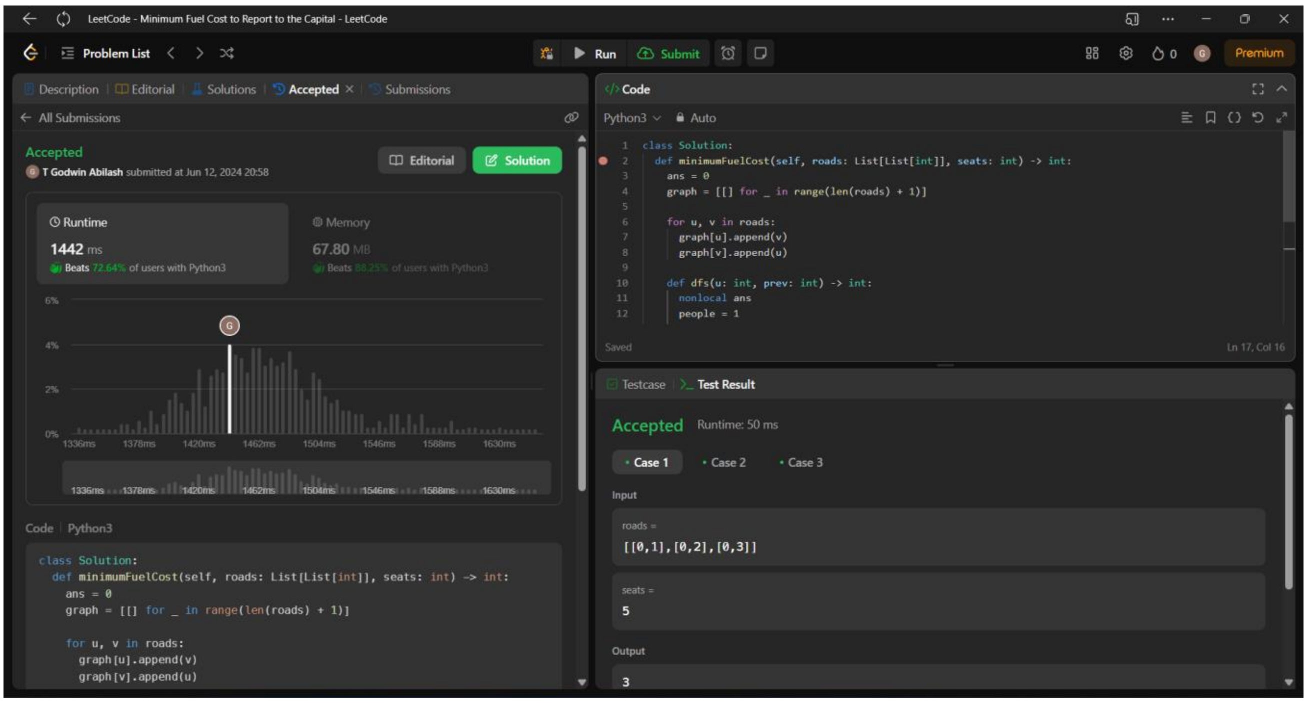
Time Complexity: O(n2)

9. Minimum Fuel Cost to Report to the Capital

Code:

```
class Solution:
  def minimumFuelCost(self, roads: List[List[int]], seats: int) -> int:
    ans = 0
    graph = [[] for _ in range(len(roads) + 1)]
    for u, v in roads:
      graph[u].append(v)
      graph[v].append(u)
    def dfs(u: int, prev: int) -> int:
      nonlocal ans
      people = 1
      for v in graph[u]:
        if v == prev:
          continue
        people += dfs(v, u)
      if u > 0:
        ans += int(math.ceil(people / seats))
      return people
    dfs(0, -1)
    return ans
```

Screenshot:



LeetCode - Minimum Fuel Cost to Report to the Capital - LeetCode

Accepted

T Godwin Abilash submitted at Jun 12, 2024 20:58

Runtime 1442 ms Beats 72.64% of users with Python3

Memory 67.80 MB Beats 88.25% of users with Python3

Code | Python3

```
class Solution:
  def minimumFuelCost(self, roads: List[List[int]], seats: int) -> int:
    ans = 0
    graph = [[] for _ in range(len(roads) + 1)]

    for u, v in roads:
      graph[u].append(v)
      graph[v].append(u)
```

```
class Solution:
  def minimumFuelCost(self, roads: List[List[int]], seats: int) -> int:
    ans = 0
    graph = [[] for _ in range(len(roads) + 1)]

    for u, v in roads:
      graph[u].append(v)
      graph[v].append(u)

    def dfs(u: int, prev: int) -> int:
      nonlocal ans
      people = 1
```

Saved Ln 17, Col 16

Accepted Runtime: 50 ms

Case 1 · Case 2 · Case 3

Input

roads =
[[0,1],[0,2],[0,3]]

seats =
5

Output

3

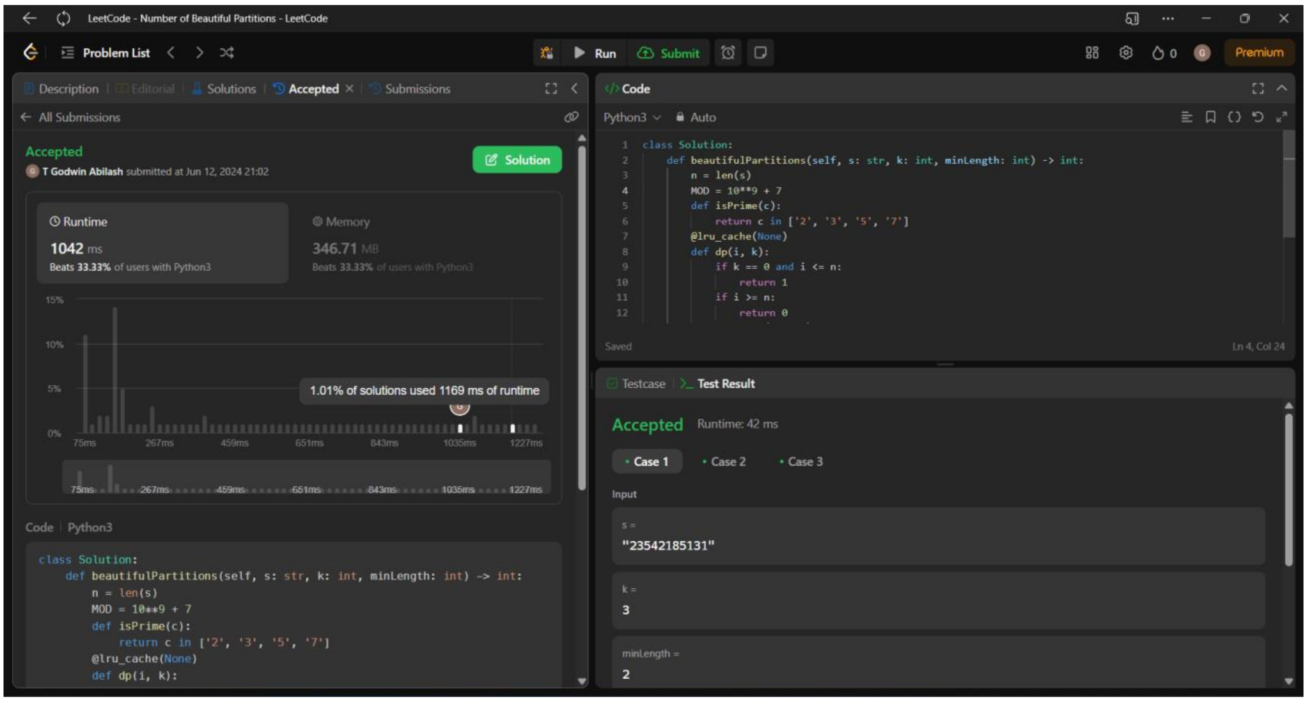
Time Complexity: O(n)

10. Number of Beautiful Partitions

Code:

```
class Solution:
    def beautifulPartitions(self, s: str, k: int, minLength: int) -> int:
        n = len(s)
        MOD = 10**9 + 7
        def isPrime(c):
            return c in ['2', '3', '5', '7']
        @lru_cache(None)
        def dp(i, k):
            if k == 0 and i <= n:
                return 1
            if i >= n:
                return 0
            ans = dp(i+1, k)  # Skip
            if isPrime(s[i]) and not isPrime(s[i-1]):  # Split
                ans += dp(i+minLength, k-1)
            return ans % MOD
        if not isPrime(s[0]) or isPrime(s[-1]): return 0
        return dp(minLength, k-1)
```

Screenshot:



Accepted
T Godwin Abilash submitted at Jun 12, 2024 21:02

Runtime
1042 ms
Beats 33.33% of users with Python3

Memory
346.71 MB
Beats 33.33% of users with Python3

1.01% of solutions used 1169 ms of runtime

Code | Python3

```
class Solution:
    def beautifulPartitions(self, s: str, k: int, minLength: int) -> int:
        n = len(s)
        MOD = 10**9 + 7
        def isPrime(c):
            return c in ['2', '3', '5', '7']
        @lru_cache(None)
        def dp(i, k):
```

```
class Solution:
    def beautifulPartitions(self, s: str, k: int, minLength: int) -> int:
        n = len(s)
        MOD = 10**9 + 7
        def isPrime(c):
            return c in ['2', '3', '5', '7']
        @lru_cache(None)
        def dp(i, k):
            if k == 0 and i <= n:
                return 1
            if i >= n:
                return 0
```

Saved
Ln 4, Col 24

Testcase | Test Result

Accepted Runtime: 42 ms

Case 1 · Case 2 · Case 3

Input

s =
"23542185131"

k =
3

minLength =
2

Time Complexity: O(n)