Python Leetcode Cheat Sheet

The Average Case assumes parameters generated uniformly at random.

Internally, a list is represented as an array; the largest costs come from growing beyond the current allocation size (because everything must move), or from inserting or deleting somewhere near the beginning (because everything after that must move). If you need to add/remove at both ends, consider using a collections.deque instead.

Operation	Average Case	Amortized Worst Case
Copy	O(n)	O(n)
Append[1]	O(1)	O(1)
Pop last	O(1)	O(1)
Pop intermediate[2]	O(n)	O(n)
Insert	O(n)	O(n)
Get Item	O(1)	O(1)
Set Item	O(1)	O(1)
Delete Item	O(n)	O(n)
Iteration	O(n)	O(n)
Get Slice	O(k)	O(k)
Del Slice	O(n)	O(n)
Set Slice	O(k+n)	O(k+n)
Extend[1]	O(k)	O(k)
Sort	O(n log n)	O(n log n)
Multiply	O(nk)	O(nk)
x in s	O(n)	
min(s), max(s)	O(n)	
Get Length	O(1)	O(1)

Python LeetCode Cheat Sheet is an invaluable resource for anyone looking to excel in coding interviews, especially those focusing on algorithm and data structure challenges. LeetCode, a popular online platform, offers a plethora of coding problems that range from easy to extremely difficult. Python, known for its simplicity and readability, is a favored language among developers. This article will provide a comprehensive cheat sheet, covering essential algorithms, data structures, and tips for solving LeetCode problems using Python.

Understanding LeetCode Problems

Before diving into the specifics of Python solutions, it’s crucial to understand the types of problems you will encounter on LeetCode. They primarily fall into several categories:

- **Array and String Manipulation:** Problems that require you to handle lists and strings effectively.
- **Linked Lists:** Involves operations on singly or doubly linked lists.
- **Trees and Graphs:** Focuses on traversing and manipulating tree and graph structures.

- **Dynamic Programming:** Problems that require breaking down tasks into simpler sub-problems.
- **Backtracking:** Problems that involve exploring all possible solutions to find the best one.
- **Sorting and Searching:** Includes algorithms to efficiently sort and search data.

Essential Python Concepts for LeetCode

To solve LeetCode problems effectively, you should be familiar with several Python concepts and built-in functions:

1. Basic Data Structures

Python provides built-in data structures that can be incredibly useful:

- Lists: Dynamic arrays that can hold different types of data.
- Dictionaries: Hash maps that allow for quick lookups.
- Sets: Unordered collections of unique elements.
- Tuples: Immutable sequences that can be used as keys in dictionaries.

2. Common Algorithms

Familiarity with common algorithms is key to solving problems quickly:

- Sorting Algorithms: Quick sort, merge sort, and built-in `sorted()` function.
- Searching Algorithms: Binary search for efficient lookups in sorted arrays.
- Graph Traversal: Depth-first search (DFS) and breadth-first search (BFS) algorithms.

3. Recursion and Backtracking

Understanding recursion is vital, particularly for tree and graph problems. Backtracking is a technique used to solve problems incrementally, abandoning paths that lead to dead ends.

Python Code Patterns for LeetCode

Recognizing common coding patterns can significantly reduce the time taken to solve problems. Here are some frequently encountered patterns along with sample code:

1. Two Pointers Technique

This technique is useful for problems involving arrays or linked lists where you need to find pairs or check for conditions.

```python
def two_sum(nums, target):
left, right = 0, len(nums) - 1
while left < right:
current_sum = nums[left] + nums[right]
if current_sum == target:
return [left, right]
elif current_sum < target:
left += 1
else:
right -= 1
```

2. Sliding Window Technique

The sliding window technique is ideal for problems involving substrings or contiguous subarrays.

```python
def length_of_longest_substring(s):
char_index_map = {}
left = max_length = 0
for right in range(len(s)):
if s[right] in char_index_map:
left = max(left, char_index_map[s[right]] + 1)
char_index_map[s[right]] = right
max_length = max(max_length, right - left + 1)
return max_length
```

3. Depth-First Search (DFS)

DFS is commonly used for tree and graph traversal.

```python
def dfs(node):
if not node:
return
print(node.val)
dfs(node.left)
dfs(node.right)
```

4. Breadth-First Search (BFS)

BFS is another traversal method, often used in graph problems.

```python
from collections import deque

def bfs(root):
queue = deque([root])
while queue:
node = queue.popleft()
print(node.val)
if node.left:
queue.append(node.left)
if node.right:
queue.append(node.right)
```

Dynamic Programming Tips

Dynamic programming (DP) is a powerful technique for solving optimization problems. Here are some tips to effectively use DP in LeetCode challenges:

1. Identify Overlapping Subproblems

Determine if the problem can be broken down into smaller, overlapping subproblems. If yes, you can use DP to store the results of these subproblems.

2. Define the State

Clearly define what each state represents in your DP solution. This often involves creating a DP table or array.

3. Establish the Recurrence Relation

Formulate a way to compute the current state based on previous states. This is where the bulk of your logic will lie.

4. Optimize Space Complexity

If possible, reduce the space complexity of your solution by only storing necessary states.

Common LeetCode Problems and Their Solutions

Here are a few common LeetCode problems along with their Python solutions:

1. Reverse a Linked List

This classic problem tests your understanding of linked lists.

```python
class ListNode:
def __init__(self, val=0, next=None):
self.val = val
self.next = next

def reverse_list(head):
prev = None
current = head
while current:
next_temp = current.next
current.next = prev
prev = current
current = next_temp
return prev
```

2. Merge Two Sorted Lists

Merging two sorted linked lists is a common interview question.

```python
def merge_two_lists(l1, l2):
dummy = ListNode()
```

```
tail = dummy
while l1 and l2:
if l1.val < l2.val:
tail.next = l1
l1 = l1.next
else:
tail.next = l2
l2 = l2.next
tail = tail.next
tail.next = l1 or l2
return dummy.next
```

3. Climbing Stairs

This dynamic programming problem involves calculating the number of ways to climb stairs.

```python
def climb_stairs(n):
if n <= 1:
return 1
dp = [0] (n + 1)
dp[0], dp[1] = 1, 1
for i in range(2, n + 1):
dp[i] = dp[i - 1] + dp[i - 2]
return dp[n]
```

Final Thoughts

Utilizing a **Python LeetCode cheat sheet** can dramatically improve your efficiency and effectiveness in tackling coding challenges. By mastering the language's built-in features, understanding common algorithms and data structures, and recognizing problem-solving patterns, you'll be well on your way to acing your coding interviews. Practice makes perfect, so take the time to solve various problems and refine your skills continuously. Happy coding!

Frequently Asked Questions

What is a Python LeetCode cheat sheet?

A Python LeetCode cheat sheet is a concise reference guide that includes commonly used algorithms, data structures, and coding patterns that can help solve problems on the LeetCode platform.

Where can I find a good Python LeetCode cheat sheet?

You can find good Python LeetCode cheat sheets on various programming blogs, GitHub repositories, and educational websites dedicated to coding and algorithm preparation.

What key topics are typically covered in a Python LeetCode cheat sheet?

Key topics often include array manipulation, string processing, dynamic programming, tree and graph algorithms, sorting and searching techniques, and commonly used libraries like collections and itertools.

How can a cheat sheet improve my performance on LeetCode?

A cheat sheet can improve performance by providing quick access to essential algorithms and coding patterns, reducing the time spent recalling syntax, and helping to identify the best approach for solving specific types of problems.

Is it advisable to rely solely on a cheat sheet when preparing for coding interviews?

While a cheat sheet is a helpful resource, it's important not to rely solely on it. Understanding the underlying concepts and practicing problems independently is crucial for effective preparation for coding interviews.

Python Leetcode Cheat Sheet

What does colon equal (:=) in Python mean? - Stack Overflow
Mar 21, 2023 · In Python this is simply =. To translate this pseudocode into Python you would need to know the data structures being referenced, and a bit more of the algorithm implementation. ...

What does asterisk * mean in Python? - Stack Overflow
What does asterisk * mean in Python? [duplicate] Asked 16 years, 7 months ago Modified 1 year, 6 months ago Viewed 319k times

What does the "at" (@) symbol do in Python? - Stack Overflow
Jun 17, 2011 · 96 What does the “at” (@) symbol do in Python? @ symbol is a syntactic sugar python provides to utilize decorator, to paraphrase the question, It's exactly about what does ...

Is there a "not equal" operator in Python? - Stack Overflow

Jun 16, 2012 · 1 You can use the != operator to check for inequality. Moreover in Python 2 there was <> operator which used to do the same thing, but it has been deprecated in Python 3.

Using or in if statement (Python) - Stack Overflow
Using or in if statement (Python) [duplicate] Asked 7 years, 6 months ago Modified 8 months ago Viewed 149k times

python - What is the purpose of the -m switch? - Stack Overflow
Python 2.4 adds the command line switch -m to allow modules to be located using the Python module namespace for execution as scripts. The motivating examples were standard library …

What is Python's equivalent of && (logical-and) in an if-statement?
Mar 21, 2010 · There is no bitwise negation in Python (just the bitwise inverse operator ~ - but that is not equivalent to not). See also 6.6. Unary arithmetic and bitwise/binary operations and 6.7. …

syntax - What do >> and <
Apr 3, 2014 · 15 The other case involving print >>obj, "Hello World" is the "print chevron" syntax for the print statement in Python 2 (removed in Python 3, replaced by the file argument of the print() …

python - Is there a difference between "==" and "is"? - Stack …
Since is for comparing objects and since in Python 3+ every variable such as string interpret as an object, let's see what happened in above paragraphs. In python there is id function that shows a …

*python - What does ** (double star/asterisk) and * (star/asterisk) do …*
Aug 31, 2008 · A Python dict, semantically used for keyword argument passing, is arbitrarily ordered. However, in Python 3.6+, keyword arguments are guaranteed to remember insertion …

What does colon equal (:=) in Python mean? - Stack Overflow
Mar 21, 2023 · In Python this is simply =. To translate this pseudocode into Python you would need to know the data structures being referenced, and a bit more of the algorithm …

*What does asterisk * mean in Python? - Stack Overflow*
*What does asterisk * mean in Python? [duplicate] Asked 16 years, 7 months ago Modified 1 year, 6 months ago Viewed 319k times*

What does the "at" (@) symbol do in Python? - Stack Overflow
Jun 17, 2011 · 96 What does the “at” (@) symbol do in Python? @ symbol is a syntactic sugar python provides to utilize decorator, to paraphrase the question, It's exactly about what does …

Is there a "not equal" operator in Python? - Stack Overflow
Jun 16, 2012 · 1 You can use the != operator to check for inequality. Moreover in Python 2 there was <> operator which used to do the same thing, but it has been deprecated in Python 3.

Using or in if statement (Python) - Stack Overflow
Using or in if statement (Python) [duplicate] Asked 7 years, 6 months ago Modified 8 months ago Viewed 149k times

python - What is the purpose of the -m switch? - Stack Overflow
Python 2.4 adds the command line switch -m to allow modules to be located using the Python module namespace for execution as scripts. The motivating examples were standard library …

What is Python's equivalent of && (logical-and) in an if-statement?
Mar 21, 2010 · There is no bitwise negation in Python (just the bitwise inverse operator ~ - but that is not equivalent to not). See also 6.6. Unary arithmetic and bitwise/binary operations and …

syntax - What do >> and <
Apr 3, 2014 · 15 The other case involving print >>obj, "Hello World" is the "print chevron" syntax for the print statement in Python 2 (removed in Python 3, replaced by the file argument of the …

python - Is there a difference between "==" and "is"? - Stack …
Since is for comparing objects and since in Python 3+ every variable such as string interpret as an object, let's see what happened in above paragraphs. In python there is id function that shows …

python - What does ** (double star/asterisk) and * (star/asterisk) …
Aug 31, 2008 · A Python dict, semantically used for keyword argument passing, is arbitrarily ordered. However, in Python 3.6+, keyword arguments are guaranteed to remember insertion …

Unlock your coding potential with our Python LeetCode cheat sheet! Get essential tips and tricks for mastering algorithms. Learn more to ace your coding interviews!

Back to Home