

Data Structures & Algorithms Cheat Sheet

Thomas Monson

Essential Patterns

Dynamic Programming

Optimal substructure \implies divide and conquer

Optimal substructure + greedy choice \implies greedy

Optimal substructure + overlapping subproblems
 \implies dynamic programming

Would it be helpful to rephrase a problem in order to more easily define its subproblems?

Given an integer array, return the length of the longest strictly increasing subsequence (LIS).

- \equiv Return the length of the LIS of an array `a` of length n .
- \equiv Return the length of the LIS of `a[0:n]`.

The LIS of `a` must have some first element. If this is the i th element, then the LIS of `a` is equal to the LIS of `a[i:]`, where `a[i]` is the first element of the sequence.

Let `dp[i]` be the length of the LIS of `a[i:]`, where `a[i]` is the first element of the sequence. Return `max(dp)`.

`@functools.lru_cache`

Searching

Here's some code for a binary search:

```
def binary_search(nums: list[int], target: int) -> int:
    left, right = 0, len(nums) - 1
    while left <= right:
        mid = (left + right) // 2
        if nums[mid] < target:
            left = mid + 1
        elif nums[mid] > target:
            right = mid - 1
        else:
            return mid
    return -1
```

bisect (binary search)

Sorting

Do you need to sort items according to a custom scheme?

- `functools.cmp_to_key`
- Create class and define dunder methods `__lt__`, `__gt__`, `__le__`, `__ge__`, `__eq__`, `__ne__`

Other Stuff

- Helper method recursion (parameter or nonlocal)
- Kadane's algorithm (maximum subarray)
- Knapsack problem (combinatorial optimization)
- Sweep line algorithm (convex hull)
- Backtracking (DFS)
- Sliding window
- Topological sorting (scheduling, Kahn's algorithm)
- LRU Cache (hash map + DLL, `OrderedDict`)
- Monotonic stack
- Union-find

Useful Python Constructs

- Would it be helpful to count items in a collection?
 \implies `Counter` creates a dictionary of the form `{element: count}`
- `defaultdict`
- `itertools.combinations`, `itertools.permutations`
- `re` (regex)
- `ord(char)` (ASCII)
- `enumerate` \rightarrow `count`, `value`

Other

- The fastest way to reverse a list is to use the “Martian smiley” `[::-1]`
- DFS \rightarrow stack (recursion) \rightarrow LIFO
- BFS \rightarrow queue (iteration) \rightarrow FIFO
- Online tests: have a Python scratchpad open, spam the “Run Tests” button (EAFP > LBYL)
- Number of subarrays of array of size n : $\frac{n(n+1)}{2}$
- Python is pass-by-assignment
 - Immutable objects are pass-by-value
 - Mutable objects are pass-by-reference
 - You can rebind the variable in the inner scope, but the outer scope will remain unchanged

Potentially Useful Algorithms

- Rabin-Karp (string-searching, uses a rolling hash to make approximate comparisons between substring hash and target hash, makes exact comparison if hashes match)
- Kruskal's algorithm and Prim's algorithm (minimum spanning tree)