# sQ811 HASHMAP

Hashmap in python, you can use a dict

Dict = {}

>>> d = {'key':'value'}

>>> print(d)

{'key': 'value'}

>>> d['mynewkey'] = 'mynewvalue'

>>> print(d)

{'mynewkey': 'mynewvalue', 'key': 'value'}

SUBSTRING in python

In general, everything before, or starting from and including the first.

>>> x = "Hello World!"

>>> x[2:]

'llo World!'

>>> x[:2]

'He'

>>> x[:-2]

'Hello Worl'

>>> x[-2:]

'd!'

>>> x[2:-2]

'llo Worl'

# Q763 Partition Labels

String loc and substring, with rfind and rindex, which find the last occurrence of a substring

# Q416 Battleships, DFS

# Q807 Max Increase to keep city sky

Just 2D array iteration

range(stop)

* stop: Number of integers (whole numbers) to generate, starting from zero. eg. range(3) == [0, 1, 2].

range([start], stop[, step])

* start: Starting number of the sequence.
* stop: Generate numbers up to, but not including this number.
* step: Difference between each number in the sequence.
* def maxIncreaseKeepingSkyline(self, grid):
* row, col = map(max, grid), map(max, zip(\*grid))
* return sum(min(i, j) **for** i **in** row **for** j **in** col) - sum(map(sum, grid))

Expression oriented functions of Python provides are:

1. map(aFunction, aSequence)
2. filter(aFunction, aSequence)
3. reduce(aFunction, aSequence)
4. lambda
5. list comprehension

# Q344 Reverse String

# Extended Slices

For example, you can now easily extract the elements of a list that have even 1indexes:

>>> L = range(10)

>>> L[::2]

[0, 2, 4, 6, 8]

Negative values also work to make a copy of the same list in reverse order:

>>> L[::-1]

[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

# Q338 Counting Bits Dynamic programming problem

**Index :** 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

**num :** **0 1 1 2** 1 2 2 3 1 2 2 3 **2 3 3 4 1 2 2 3 2 3 3 4**

dp[0] = 0;

dp[1] = dp[1-1] + 1;

dp[2] = dp[2-2] + 1;

dp[3] = dp[3-2] +1;

dp[4] = dp[4-4] + 1;

dp[5] = dp[5-4] + 1;

dp[6] = dp[6-4] + 1;

dp[7] = dp[7-4] + 1;

dp[8] = dp[8-8] + 1;

# 191 Number of 1 Bits

bin(n).count('1')

# 791 Custom Sort String

Character array with count method,

Run time O(N^2)

You can also use count the number of times a character appears.

# 442 Find All Duplicates in an Array

Using the input array as a hash function, by changing the value to negative to indicate that this spot has been visited.

# 406 Queue Reconstruciton by height

Dynamic programming, find the position for the shortest person first,

Then second shortest.

Second solution,

Hash Map, hash on height,

# Q496 Next Greater Element Stack!!!

Solve by creating a dict for each value, since there is no duplicates, and 1 is subset of 2.

Actually, use stack!!!!

Used array to build a stack structure, and array [-1] is the top of the stack, array has append and pop()

diction, st = {} , []

for i in nums:

if(len(st) == 0 ):

st.append(i)

elif(i < st[-1]):

st.append(i)

else:

while st and st[-1] < i:

diction[st.pop()] = i

st.append(i)

# Q75 Sort Colors Dutch partitioning problem

The basic idea of quick sort

Sort 0 ,1 ,2 counting sort is 2n

Need a n solution.

Use 3 different point to classfy the unknown items to the correct posiotn, using swap

# Q162 find peak element binary search!!!

First define left and right, which is 0 and length -1

Then depending on the condition left= mid +1 or right = mid -1

**if** an element(**not** the right-most one) **is** smaller than its right neighbor, **then** there must be a peak element **on** its right, because the elements **on** its right **is** either

1. always increasing -> the right-most element **is** the peak

2. always decreasing -> the left-most element **is** the peak

3. first increasing **then** decreasing -> the pivot point **is** the peak

4. first decreasing **then** increasing -> the left-most element **is** the peak

# Q240 Search a 2D Matrix 2

First solution, use binary search, left, right mid for each solution, then this is n\*n

But, the better solution is to go through column and row at the same time.

Suppose we want to search for 12. We first initialize r = 0 and c = 4. We compare 12 with matrix[r][c] = matrix[0][4] = 15 and 12 < 15, so 12 cannot appear in the column of 15since all elements below 15 are not less than 15. Thus, we decrease c by 1 and reduce the search range by a column. Now we compare 12 with matrix[r][c] = matrix[0][3] = 11 and 12 > 11, so 12 cannot appear in the row of 11 since all elements left to 11 are not greater than 11. Thus, we increase r by 1 and reduce the search range by a row.

# Q49 Group Anagrams

The hashmap in python, dict can have tuples as keys, which means, (a,b,c) can be a key

And the tuple() function tuple('abc') returns ('a', 'b', 'c') and tuple([1, 2, 3]) returns (1, 2, 3).

**tuple**([iterable])

\*\*\* you have to sort the strings first !

Following is the syntax for **get()** method −

dict.get(key, default = None)

## **Parameters**

* **key** − This is the Key to be searched in the dictionary.
* **default** − This is the Value to be returned in case key does not exist.

# 300 Longest increasing subsequences DP and binary search

Using dp, for each new added in value, check all the ones before it, and if this value is larger than anyone before it ,find the max LIS then +1

# 136 single number

dic = {}

**for** num **in** nums:

dic[num] = dic.get(num, 0)+1

**for** key, val **in** dic.items():

**if** val == 1:

**return** key

Hash map, iterate with key and vals

Or you can use xor and the only one number with one copy will be the result.

# 104 Maximum Depth of Binary Tree

When you are doing recursion, you need to use self.

if(root.left != None):

dpl += self.maxDepth(root.left)

if(root.right != None):

dpr += self.maxDepth(root.right)