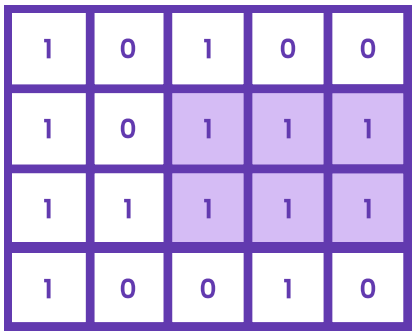
Stacks

Assignment Solutions

PW SKILLS

Q1. Given a rows x cols binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return its area.

Example 1:



Input: matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]
Output: 6
Explanation: The maximal rectangle is shown in the above picture.

Example 2:
Input: matrix = [["0"]]
Output: 0

Example 3:
Input: matrix = [["1"]]
Output: 1

Solution :

Code : ASS_Code1.java

Output :

```
The maximum area is : 2
```

Approach :

- Pick one row.
- Do summation of each index till that row
  i) if any index value is 0 then put 0 else previous summation + 1
- Pass this array to get max area (Largest Rectangle in Histogram that we discussed in lecture.)
- Update max area.

Largest Rectangle in Histogram

- Max area will always have at least one full bar height on any index
- Find the largest rectangle including each bar one by one.
  a) For each bar, We have to find it's left limit & right limit (to know the maximum width)

b) Find it's left limit (where we find any index's value is smaller than current index in left side array of curr index)

c) Find it's right limit (where we find any index's value is smaller than current index in right side array of curr index

- Take the maximum of all the max areas found by each bar.
- calculate area = width * height where width = right limit - left limit + 1 and height = curr index's value
- Update max area & return it.

Q2. Given an encoded string, return its decoded string. The encoding rule is: k[encoded_string], where the encoded_string inside the square brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer.
You may assume that the input string is always valid; there are no extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there will not be input like 3a or 2[4].

Example 1:
Input: s = "3[a]2[bc]"
Output: "aaabcbc"

Example 2:
Input: s = "3[a2[c]]"
Output: "accaccacc"

Example 3:
Input: s = "2[abc]3[cd]ef"
Output: "abcabccdcdcdef"

Solution :

Code : ASS_Code2.java

Output :

```
Enter a string: 3[abc4[ds]]
The decoded string is : abcdsdsdsdsabcdsdsdsdsabcdsdsdsds
```

Approach :

- If string does not have inner substring like this 3[a5[cd]] then it can be solved easily.(simple iteration)
- In some cases, we can have an inner substring as I mentioned above, then it is best to solve it with a stack. Solve inner substring first.(Iterative approach)
- Insert the character in the stack until you find ']' char.
- If you find ']' char then pop the character until you find '[', This is how you can get the substring.
- Remove the '[' character.
- Find the number k, number can be in single digit, two digits, and so on.

- Put back the substring k times in the stack.
- At last take the result in one char array because stack format will not be in string format.
- Return the result.

Q3. You are keeping the scores for a baseball game with strange rules. At the beginning of the game, you start with an empty record.

You are given a list of strings operations, where operations[i] is the ith operation you must apply to the record and is one of the following:

An integer x.
Record a new score of x.
'+'.
Record a new score that is the sum of the previous two scores.
'D'.
Record a new score that is the double of the previous score.
'C'.
Invalidate the previous score, removing it from the record.
Return the sum of all the scores on the record after applying all the operations.

Example 1:
Input: ops = ["5","2","C","D","+"]
Output: 30
Explanation:
"5" - Add 5 to the record, record is now [5].
"2" - Add 2 to the record, record is now [5, 2].
"C" - Invalidate and remove the previous score, record is now [5].
"D" - Add 2 * 5 = 10 to the record, record is now [5, 10].
"+" - Add 5 + 10 = 15 to the record, record is now [5, 10, 15].
The total sum is 5 + 10 + 15 = 30.

Example 2:
Input: ops = ["5","-2","4","C","D","9","+","+"]
Output: 27
Explanation:
"5" - Add 5 to the record, record is now [5].
"-2" - Add -2 to the record, record is now [5, -2].
"4" - Add 4 to the record, record is now [5, -2, 4].
"C" - Invalidate and remove the previous score, record is now [5, -2].
"D" - Add 2 * -2 = -4 to the record, record is now [5, -2, -4].
"9" - Add 9 to the record, record is now [5, -2, -4, 9].
"+" - Add -4 + 9 = 5 to the record, record is now [5, -2, -4, 9, 5].
"+" - Add 9 + 5 = 14 to the record, record is now [5, -2, -4, 9, 5, 14].
The total sum is 5 + -2 + -4 + 9 + 5 + 14 = 27.

Example 3:
Input: ops = ["1","C"]
Output: 0
Explanation:
"1" - Add 1 to the record, record is now [1].
"C" - Invalidate and remove the previous score, record is now [].
Since the record is empty, the total sum is 0.

Solution :

Code : ASS_Code3.java

Output :

```
The score is : 30
```

Approach :

- We use a stack to solve this problem and we handle the following cases:
  1. If ops[i]=="C"
- we decrease the sum by the popped element.
  2. Else if ops[i]=="D"
- we increase the sum by double of the top elemet of the stack and then also push that number into the stack
  3. Else if ops[i]=="+"
- we remove the top element of stack and store it in a variable x
- Then we again remove the new top element of the stack and store it in a variable y.
- Now we increase the sum by x+y.
- Then we push y first then x then x+y into the stack
  4. ELSE (if its a number string)
- we push the element into the stack.

Q4. We are given an array of asteroids of integers representing asteroids in a row.For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed.
Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are the same size, both will explode. Two asteroids moving in the same direction will never meet.

Example 1:
Input: asteroids = [5,10,-5]
Output: [5,10]
Explanation: The 10 and -5 collide resulting in 10. The 5 and 10 never collide.

Example 2:
Input: asteroids = [8,-8]
Output: []

Explanation: The 8 and -8 collide exploding each other.

Example 3:
Input: asteroids = [10,2,-5]
Output: [10]
Explanation: The 2 and -5 collide resulting in -5. The 10 and -5 collide resulting in 10.

Solution :

Code : ASS_Code4.java

Output :

```
The desired output is :
5 10
```

Approach :

- It will collide only when Last element is negative and the 2nd last element is positive.
- Vice-versa of the above statement will not cause collision because if the last element is positive then it will go towards the right side and if 2nd last is negative then it will go towards the left side and hence it'll not collide.

  e.g:-

  Case 1:- If last is +ve and 2nd last is +ve:-
  <---- 2nd last last----> Hence not collide

  Case 2:- If last is -ve and 2nd last is +ve:
  2nd last ----> <----last Hence it'll collide

Q5. Given an array of integers temperatures represents the daily temperatures, return an array answer such that answer[i] is the number of days you have to wait after the ith day to get a warmer temperature. If there is no future day for which this is possible, keep answer[i] == 0 instead.

Example 1:
Input: temperatures = [73,74,75,71,69,72,76,73]
Output: [1,1,4,2,1,1,0,0]

Example 2:
Input: temperatures = [30,40,50,60]
Output: [1,1,1,0]

Example 3:
Input: temperatures = [30,60,90]
Output: [1,1,0]

Solution :

Code : ASS_Code5.java

Output :

```
The desired output is :
3 2 1 2 1 1 0
```

Approach :

- For a temperature at index i, we need to find the slightly warmer temperature (at position j) than the current temperature for 0 <= i < j < n. Which means we need to find the next greater element for the current element at index i and if not found then we place 0 in that place.
- Next Greater Element is a standard problem that is done by using stack and here we just have a slight variation in it. Instead of finding the next greater element, we need to find the distance between the current element (temperature) and the next greater element which can be easily done.
- We will keep a stack and we will start iterating from the end of the temperature array(n-1).
- Now, our main objective is to keep the candidate temperatures on the stack that can be a next greater temperature for the current index.
- Now, for a current temperature at index i, we will see the top of the stack and check whether it is greater than the current element or not. We do this because stack is a LIFO and we came from right, so the top of the stack will be the latest element/temperature, that would have been pushed into the stack from the right of current index i so here can be two cases:
- Case 1: When this temperature at top of stack is less than the current temperature, then this can never be a next greater element of current temperature or all the other temperatures before current temperature. Because for temperatures before current temperature (at position < i), temperature at current index can be their candidate for the next greater element. Hence we will simply pop the top of the stack out until we find an element greater than the current index or until the stack gets empty (This means no greater element on the right).
- Case 2: When this temperature at top of the stack is greater than the current then we will simply consider this element as our next greater element and set the distance between the current temperature and next greater element/temperature as our answer for the current index.
- After all this is done, then we will have either 0 or the distance between the current temp and next greater temp in the answer vector. Then we will simply push our current element into the stack as now this temp can be a candidate to become the next greater element for the temperatures before this temperature. And then move to the left Note: We are referring to the next greater element as “nge”.