

918. Maximum Sum Circular Subarray

Attempted

Medium Topics Companies Hint

Given a **circular integer array** `nums` of length `n`, return the *maximum possible sum of a non-empty **subarray*** of `nums`.

A **circular array** means the end of the array connects to the beginning of the array. Formally, the next element of `nums[i]` is `nums[(i + 1) % n]` and the previous element of `nums[i]` is `nums[(i - 1 + n) % n]`.

A **subarray** may only include each element of the fixed buffer `nums` at most once. Formally, for a subarray `nums[i], nums[i + 1], ..., nums[j]`, there does not exist `i <= k1, k2 <= j` with `k1 % n == k2 % n`.

Example 1:

Input: `nums = [1,-2,3,-2]`
Output: 3
Explanation: Subarray [3] has maximum sum 3.

Example 2:

Input: `nums = [5,-3,5]`
Output: 10
Explanation: Subarray [5,5] has maximum sum 5 + 5 = 10.

Example 3:

Input: `nums = [-3,-2,-3]`
Output: -2
Explanation: Subarray [-2] has maximum sum -2.

Constraints:

`n == nums.length`

`1 <= n <= 104`

6.8K 79

110 Online

Code

C++ Auto

```
1 //Sanjay Chandagani VU21CSEN00300372
2 class Solution {
3 public:
4     int kadane(vector<int>& nums) {
5         int currSum = nums[0];
6         int maxSum = nums[0];
7         for(int i = 1; i < nums.size(); i++) {
8             currSum = max(nums[i], currSum + nums[i]);
9             maxSum = max(maxSum, currSum);
10        }
11        return maxSum;
12    }
13
14    int maxSubarraySumCircular(vector<int>& nums) {
15        // If all numbers are negative, return maximum element
16        bool allNegative = true;
17        for(int num : nums) {
18            if(num >= 0) {
19                allNegative = false;
20                break;
21            }
22        }
23        if(allNegative) {
24            return *max_element(nums.begin(), nums.end());
25        }
26        return max(kadane(nums), kadane(nums) - min_element(nums.begin(), nums.end(), [](int a, int b) { return a < b; }));
27    }
28 }
```

Ln 18, Col 26 Saved



Run

Submit

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

`nums =`
`[1,-2,3,-2]`

Output

3

 Description |  Editorial |  Solutions |  Submissions

936. Stamping The Sequence

Hard

Topics

Companies

You are given two strings `stamp` and `target`. Initially, there is a string `s` of length `target.length` with all `s[i] == '?'`.

In one turn, you can place `stamp` over `s` and replace every letter in the `s` with the corresponding letter from `stamp`.

- For example, if `stamp = "abc"` and `target = "abcba"`, then `s` is `"?????"` initially. In one turn you can:
 - place `stamp` at index `0` of `s` to obtain `"abc??"`,
 - place `stamp` at index `1` of `s` to obtain `"?abc?"`, or
 - place `stamp` at index `2` of `s` to obtain `"??abc"`.

Note that `stamp` must be fully contained in the boundaries of `s` in order to stamp (i.e., you cannot place `stamp` at index `3` of `s`).

We want to convert `s` to `target` using **at most** `10 * target.length` turns.

Return an array of the index of the left-most letter being stamped at each turn. If we cannot obtain `target` from `s` within `10 * target.length` turns, return an empty array.

Example 1:

Input: `stamp = "abc", target = "ababc"`
Output: `[0,2]`
Explanation: Initially `s = "?????"`.

- Place stamp at index `0` to get `"abc??"`.
- Place stamp at index `2` to get `"ababc"`.

`[1,0,2]` would also be accepted as an answer, as well as some other answers.

Example 2:

Input: `stamp = "abca", target = "aabcaca"`
Output: `[2,0,1]`

</> Code

C++

Auto

```
1 //Sanjay Chandagani VU21CSEN0300372
2 class Solution {
3 public:
4     bool canReplace(string& target, int pos, string& stamp) {
5         // Check if we can replace the window at pos with stamp
6         for(int i = 0; i < stamp.length(); i++) {
7             if(target[i + pos] != '?' && target[i + pos] != stamp[i]) {
8                 return false;
9             }
10        }
11        return true;
12    }
13
14    int replace(string& target, int pos, int len, int count) {
15        // Replace the window with '?' and return number of new '?' added
16        int newCount = 0;
17        for(int i = 0; i < len; i++) {
18            if(target[pos + i] != '?') {
19                target[pos + i] = '?';
20                newCount++;
21            }
22        }
23        return newCount;
24    }
25
26    vector<int> stampingSequence(string& stamp, string& target) {
27        int n = target.length();
28        int len = stamp.length();
29        int count = 0;
30        vector<int> ans;
31        while(count <= 10 * n) {
32            for(int i = 0; i <= n - len; i++) {
33                if(canReplace(target, i, stamp)) {
34                    int newCount = replace(target, i, len, count);
35                    count += newCount;
36                    ans.push_back(i);
37                }
38            }
39            if(count == n) {
40                return ans;
41            }
42        }
43        return {};
44    }
45
46 private:
47     string s;
48 };
49
50 int main() {
51     Solution sol;
52     string stamp = "abc";
53     string target = "ababc";
54     vector<int> ans = sol.stampingSequence(stamp, target);
55     for(int i = 0; i < ans.size(); i++) {
56         cout << ans[i] << " ";
57     }
58     return 0;
59 }
```

Ln 1, Col 36 | Saved

 Run  Submit

☒ Testcase |  Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

stamp =
"abc"

target =
"ababc"