Description | Editorial | Solutions | Submissions

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 <= 3 * 10^4`

6.8K | 79 | 110 Online

Code

C++ | Auto

```cpp
//Sanjay Chandagani VU21CSEN00300372
class Solution {
public:
    int kadane(vector<int>& nums) {
        int currSum = nums[0];
        int maxSum = nums[0];
        for(int i = 1; i < nums.size(); i++) {
            currSum = max(nums[i], currSum + nums[i]);
            maxSum = max(maxSum, currSum);
        }
        return maxSum;
    }

    int maxSubarraySumCircular(vector<int>& nums) {
        // If all numbers are negative, return maximum element
        bool allNegative = true;
        for(int num : nums) {
            if(num >= 0) {
                allNegative = false;
                break;
```

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"
```

1.6K | 13 | 23 Online

Code

C++ | Auto

```cpp
//Sanjay Chandagani VU21CSEN0300372
class Solution {
public:
    bool canReplace(string& target, int pos, string& stamp) {
        // Check if we can replace the window at pos with stamp
        for(int i = 0; i < stamp.length(); i++) {
            if(target[i + pos] != '?' && target[i + pos] != stamp[i]) {
                return false;
            }
        }
        return true;
    }

    int replace(string& target, int pos, int len, int count) {
        // Replace the window with '?' and return number of new '?' added
        int newCount = 0;
        for(int i = 0; i < len; i++) {
            if(target[pos + i] != '?') {
                target[pos + i] = '?';
                newCount++;
```

Ln 1, Col 36 | Saved

Run | Submit

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

stamp =

"abc"

target =

"ababc"