

## 6328. Find the Substring With Maximum Cost

[My Submissions \(/contest/biweekly-contest-101/problems/find-the-substring-with-maximum-cost/submissions/\)](#)
[Back to Contest \(/contest/biweekly-contest-101/\)](#)

You are given a string `s`, a string `chars` of **distinct** characters and an integer array `vals` of the same length as `chars`.

The **cost of the substring** is the sum of the values of each character in the substring. The cost of an empty string is considered `0`.

The **value of the character** is defined in the following way:

- If the character is not in the string `chars`, then its value is its corresponding position (**1-indexed**) in the alphabet.
  - For example, the value of 'a' is 1, the value of 'b' is 2, and so on. The value of 'z' is 26.
- Otherwise, assuming `i` is the index where the character occurs in the string `chars`, then its value is `vals[i]`.

Return the maximum cost among all substrings of the string `s`.

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

### Example 1:

**Input:** `s = "adaa", chars = "d", vals = [-1000]`

**Output:** 2

**Explanation:** The value of the characters "a" and "d" is 1 and -1000 respectively. The substring with the maximum cost is "aa" and its cost is 1 + 1 = 2. It can be proven that 2 is the maximum cost.

### Example 2:

**Input:** `s = "abc", chars = "abc", vals = [-1,-1,-1]`

**Output:** 0

**Explanation:** The value of the characters "a", "b" and "c" is -1, -1, and -1 respectively. The substring with the maximum cost is the empty substring "" and its cost is 0. It can be proven that 0 is the maximum cost.

### Constraints:

- `1 <= s.length <= 105`
- `s` consist of lowercase English letters.
- `1 <= chars.length <= 26`
- `chars` consist of **distinct** lowercase English letters.
- `vals.length == chars.length`
- `-1000 <= vals[i] <= 1000`

JavaScript



```

1 const ord = (c) => c.charCodeAt();
2 const maximumCostSubstring = (s, a, b) => {
3   let sm = new Map(), pre = [0], n = s.length, m = a.length;
4   for (let i = 0; i < m; i++) sm.set(a[i], i);
5   for (const c of s) {
6     let v = sm.has(c) ? b[sm.get(c)] : ord(c) - 96;
7     pre.push(pre[pre.length - 1] + v);
8   }
9   // pr(pre);
10  let len = pre.length, suf = Array(len).fill(Number.MIN_SAFE_INTEGER), res = Number.MIN_SAFE_INTEGER;
11  suf[len - 1] = pre[len - 1]
12  for (let i = len - 2; i >= 0; i--) suf[i] = Math.max(suf[i + 1], pre[i]);
13  // pr(suf)
14  for (let i = 0; i < len; i++) { // max value in [i+1, ...]
15    let rangeSum = suf[i] - pre[i];
16    res = Math.max(res, rangeSum);
17  }
18  return res;
19 };

```

☐ Custom Testcase

Use Example Testcases

Run

Submit

Submission Result: **Accepted** (/submissions/detail/926005687/) ⓘ More Details > (/submissions/detail/926005687/)

Share your acceptance!