Task 1:

Write a function `solution` that, given a string S of length N, returns the length of the shortest *unique* substring of S, that is, the length of the shortest word which occurs in S exactly once.

Examples:

1. Given S = "abaaba", the function should return 2. The shortest unique substring of S is "aa".

2. Given S = "zyzyzyz", the function should return 5. The shortest unique substring of S is "yzyzy". Note that there are shorter words, like "yzy", occurrences of which overlap, but they still count as multiple occurrences.

3. Given S = "aabbbabaaa", the function should return 3. All substrings of size 2 occurs in S at least twice.

Assume that:

- N is an integer within the range [1..200];
- string S is made only of lowercase letters (a-z).

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Solution:

```
function solution(S) {
  const counts = {};
  const n = S.length;

  for (let i = 0; i < n; i++) {
    for (let j = i + 1; j <= n; j++) {
      const substr = S.substring(i, j);

      if (!(substr in counts)) {
        counts[substr] = 1;
      } else {
        counts[substr]++;
      }
    }
  }

  for (let i = 1; i <= n; i++) {
    for (const substr in counts) {
      if (counts[substr] === 1 && substr.length === i) {
        return i;
      }
    }
  }

  return -1;
}
```

Task 2:

We are given a string S of length N consisting only of letters 'A' and/or 'B'. Our goal is to obtain a string in the format "A...AB...B" (all letters 'A' occur before all letters 'B') by deleting some letters from S. In particular, strings consisting only of letters 'A' or only of letters 'B' fit this format.

Write a function:

```
class Solution { public int solution(String S); }
```

that, given a string S, returns the minimum number of letters that need to be deleted from S in order to obtain a string in the above format.

Examples:

1. Given S = "BAAABAB", the function should return 2. We can obtain "AAABB" by deleting the first occurrence of 'B' and the last occurrence of 'A'.

2. Given S = "BBABAA", the function should return 3. We can delete all occurrences of 'A' or all occurrences of 'B'.

3. Given S = "AABBBB", the function should return 0. We do not have to delete any letters, because the given string is already in the expected format.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- string S is made only of the characters 'A' and/or 'B'.

Solution for task 2:

```
function solution(S) {
    let aCount = 0;
    let bCount = 0;
    let bToACount = 0;

    for (let i = 0; i < S.length; i++) {
      if (S[i] === "A") {
        aCount++;
        bToACount = Math.min(bCount, bToACount + 1);
      } else {
        bCount++;
      }
    }

    return bToACount;
  }
```

Task 3:

A company is planning N projects, numbered from 0 to N−1. Completing the K-th project will bring value V[K] to the company. For some projects there may be additional requirements - the L-th requirement states that before starting project B[L], project A[L] should be completed. There are M such requirements.

The company has enough resources for **at most two** projects to be completed. If two projects are chosen, they will be completed one by one (in sequential order) and the total value they bring to the company is the sum of their individual values. What is the highest value that a valid choice of projects can bring to the company?

Write a function:

```
function solution(V, A, B);
```

that, given array V of N integers and two arrays A and B of M integers each, returns the maximum value that the company may gain by completing at most two possible projects.

Examples:

1. Given V = [−3, 5, 7, 2, 3], A = [3, 1] and B = [2, 4], the function should return 9. This can be achieved by completing project 3 (with value 2) first and then project 2 (with value 7).

2. Given V = [1, 1, 5], A = [0, 1] and B = [2, 2], the function should return 2. This can be achieved by completing projects 0 and 1. Project 2 can not be completed as it requires both projects 0 and 1 completed before.

3. Given V = [5, 6, 6, 7, −10], A = [0, 0, 0, 1, 2, 3] and B = [1, 2, 3, 3, 1, 2], the function should return 5. The projects that are possible to be completed are 0 and 4. As project 4 would bring negative value to the company, it is optimal to do only project 0. The structure of dependencies of projects 1, 2 and 3 form a cycle, so none of them can be completed in a valid choice of projects.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- M is an integer within the range [0..100,000];
- each element of array V is an integer within the range [−1,000,000,000..1,000,000,000];
- each element of arrays A and B is an integer within the range [0..N-1];
- a project may not require itself to be completed (A[K] ≠ B[K]);
- projects' requirements do not repeat.

I don’t have accurate solutions that worked for me.