C++ (ENGLISH)

7. forduló

Google

A kategória támogatója: Google

Ismertető a feladatlaphoz

Az utolsó fordulókhoz érkeztünk, így megosztunk 1-2 fontos információt a továbbiakról:

- a versennyel kapcsolatos észrevételeket december 5-ig tudjátok velünk megosztani a szokásos helyen
- az utolsó fordulóhoz kapcsolódó megoldások november 30-án érhetők el
- a végeredményről tájékoztatás decemberben, részletek hamarosan

Sok sikert az utolsó fordulóhoz!

This round consists of algorithmic problems. Before the timer starts, please prepare your favorite C++ IDE (C++ version 17 or above is recommended). The input will be provided in a downloadable include file (less than 100KB), and you will also find a solution code template.

You are allowed to use any content that you own or find on the internet that was published before the start of the round. However, any other form of assistance is prohibited.

If for any reason you are unable to download the zip directly from the application (typically a virus killer can hang it), you can also get a Google Drive link in the first question.

`Indítás utáni csatolmányok`

1. feladat 10 pont

Color lists

You live in a village near a school. Your house had a long fence with narrow unpainted vertical planks. You found this boring, so you replaced each plank with a random colored one. Amos is a child who passes your house twice a

day in opposite directions.

The input for all tasks consists of 10 test cases. Each of them has:

> A single list of positive integers that represents the colors of the painted rails from left to right. You don't have to check the input for correctness.

You get the input in a C++ file, color_lists_inputs.cpp.inc. You have to provide a space-separated list of integers, the answers for each test case (see the input style with the "sample inputs" and a suggested template code below, you only have to implement the solution logic).

Amos reads out the rail colors when passing by to and from school, and complains if the color lists in the morning and in the afternoon are different.

You want to repaint some rails so that the color lists Amos reads out become equal. How many rails do you have to repaint at the minimum to achieve this?

Sample test case: colors = {1,2,2,3,4,1,4,1,1,2,2,2,2}

Sample answer: 3 (e.g. 1,2,2,***2***,4,1,4,1,***4***,2,2,2,***1***)

Answer for the sample inputs (test_cases_task_1_2 below) is 3 5 39

Válasz

2. feladat 10 pont

Task 2

Repainting the rails was a good plan, but is not viable, as you realized that you cannot buy the same colors as a bucket of paint. So you decide to replace some rails with the removed unpainted ones (which are not read out at all) so that the color lists Amos reads out become equal. How many rails do you have to replace at the minimum to achieve this?

Sample test case: colors = {1,2,2,3,4,1,4,1,1,2,2,2,2}
Sample answer: 6 (e.g. -,2,2,-,-,1,4,1,-,2,2,-,-)
Answer for the sample inputs (test_cases_task_1_2 below) is 6 4 35

Válaszok

3. feladat 10 pont

Task 3

While showing your plan to Amos, he starts to read out from paper, and you realize that Amos does not like saying the same color multiple times. If the same color is painted multiple times next to each other, he only says it once (unpainted rails are not taken into consideration at all), e.g. colorlist 1,2,2,-,2,-,1,1 becomes 1,2,1 when he reads it.

Fortunately, you only used 2 different colors this time (for all Task 3 test cases). How many rails at the minimum do you have to replace with an unpainted one in this case so that the two lists to and from school sound the same?

Sample test case: colors = {1,1,1,2,2,2,1,1,1,2,2,2,2,1,1,2,2,1,1,1,2,2}

Sample answer: 2 (e.g. 1,1,1,2,2,2,1,1,1,2,2,2,2,1,1,2,2,1,1,1,-,- which reads out 1,2,1,2,1,2,1)

Answer for the sample inputs (test_cases_task_3 below): 2 0 3

Common code snippets

The input file will be similar to this sample input file:

```
std::vector<std::vector<int>> test_cases_task_1_2 = {
    /* sample test cases */
    {1, 2, 2, 3, 4, 1, 4, 1, 1, 2, 2, 2, 2},
    {1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 1, 1, 1, 2, 2},
    {6, 8, 7, 4, 7, 3, 4, 6, 3, 6, 12, 12, 8, 4, 8, 6, 7, 12, 12, 7, 6, 8, 3, 12,

std::vector<std::vector<int>> test_cases_task_3 = {
    /* sample test cases */
    {1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 1, 1, 1, 2, 2},
    {6, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 6, 6, 5, 5, 6, 6, 6},
    {4, 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 4, 4, 4, 3, 3, 4, 3, 4, 4, 3, 3,
    4, 4, 3, 3, 3}};
```

You can use the *color_lists_inputs.cpp.inc* file like:

```
#include <iostream>
#include "color_lists_inputs.cpp.inc"
#include <vector>

// copying the input so it can be modified locally
int task1(std::vector<int> test_case) {
  // TODO: Task 1 solution here
  return 0;
}
```

```cpp
int task2(std::vector<int> test_case) {
  // TODO: Task 2 solution here
  return 0;
}

int task3(std::vector<int> test_case) {
  // TODO: Task 3 solution here
  return 0;
}

int main() {
  std::cout << "Solution for Task 1:" << std::endl;
  for (std::vector<int> &test_case : test_cases_task_1_2) {
    std::cout << task1(test_case) << " ";
  }
  std::cout << std::endl;
  std::cout << "Solution for Task 2:" << std::endl;
  for (std::vector<int> &test_case : test_cases_task_1_2) {
    std::cout << task2(test_case) << " ";
  }
  std::cout << std::endl;
  std::cout << "Solution for Task 3:" << std::endl;
  for (std::vector<int> &test_case : test_cases_task_3) {
    std::cout << task3(test_case) << " ";
  }
  std::cout << std::endl;
  return 0;
}
```

We've thought a bit more about this last question, if you want to do the "bonus" exercise, you can find it on the "goodbye page".

Válaszok

Megoldások beküldése