给定两个字符串str1,str2和一个目标字符串aim,判断aim是否由str1和str2交替构成。

```
#include <map>
#include <string>
#include <vector>

// 能够判断，aim是由str1和str2组成，但无法判断其是否按顺序
bool isstrCom(std::string str1, std::string str2, std::string aim) {
  int len1 = str1.size();
  int len2 = str2.size();
  int len3 = aim.size();

  if (len3 != len2 + len1) {
    return false;
  }

  std::map<char, int> map1;
  std::map<char, int> map2;

  for (int i = 0; i < len1; i++) {
    map1[str1[i]]++;
  }

  for (int i = 0; i < len2; i++) {
    map2[str2[i]]++;
  }

  bool flag1 = false;
  bool flag2 = false;

  for (int i = 0; i < len3; i++) {
    flag1 = false;
    flag2 = false;

    if (map1.count(aim[i])) {
      if (map1[aim[i]] > 0) {
        flag1 = true;
        map1[aim[i]]--;
      } else {
        // str1中字符不满足
        map1.erase(aim[i]);
        flag1 = false;
      }
    }

    if (!flag1 && map2.count(aim[i])) {
      if (map2[aim[i]] > 0) {
        flag2 = true;
        map2[aim[i]]--;
```

```
      } else {
        // str2中字符不满足
        map2.erase(aim[i]);
        flag2 = false;
      }
    }

    if (!flag1 && !flag2) {
      return false;  // str1 与 str2 中均不存在
    }
  }

  return true;
}

// 动态规划
bool isCross(std::string str1, std::string str2, std::string aim) {
  int len1 = str1.size();
  int len2 = str2.size();
  int len3 = aim.size();

  if (len3 != len1 + len2) {
    return false;
  }

  std::vector<std::vector<bool>> dp =
      std::vector<std::vector<bool>>(len1 + 1,
                                     std::vector<bool>(len2 + 1, false));
  // base case
  dp[0][0] = true;

  for (int i = 1; i <= len1; i++) {
    if (str1[i - 1] != aim[i - 1]) {
      break;
    }
    dp[i][0] = true;
  }

  for (int j = 1; j <= len2; j++) {
    if (str2[j - 1] != aim[j - 1]) {
      break;
    }

    dp[0][j] = true;
  }

  // 结算dp
  for (int i = 1; i <= len1; i++) {
    for (int j = 1; j <= len2; j++) {
      if (str1[i - 1] == aim[i + j - 1] && dp[i - 1][j] ||
          str2[j - 1] == aim[i + j - 1] && dp[i][j - 1]) {
        dp[i][j] = true;
      }
    }
```

```
    }

    return dp[len1][len2];
}
```