# משימה 1 מונופול

קוביליו נהנה מאוד לשחק במונופול. אבל משך משחק אחד הוא ארוך מאוד - מ-5 עד 7 שעות. אז, קוביליו מתחיל לחשוב על שינוי הכללים הקלאסיים. השחקנים במונופול נעים בדרך כלל בכיוון אחד ממשבצת אחת לאחרת ולאחר זמן מה הם חוזרים להתחלה, מתחילים מחדש וכן הלאה. בגרסה החדשה, התנועה תהיה כרגיל ממשבצת למשבצת אך בשונה מהגירסא הרגילה יכולות להיות מספר אפשרויות למהלך הבא. קוביליו רוצה למצוא קשתות מכוונות כאלה בין המשבצות, כך ששחקן לעולם לא יוכל לחזור למשבצת שבה הוא היה, לא משנה איך הוא זז (כמובן, לפי הכללים). באופן כזה המשחק יהיה קצר יותר.

הוא כבר התחיל להכין את הלוח החדש - הוא בחר את מספר המשבצות N (המשבצות ממוספרות מ-1 עד N) והוא עשה רשימה עם M קשתות (לכל קשת יש כיוון ואין קשת שמחבר בין משבצת לעצמה). אם יש קשת בין משבצת i למשבצת j, אז אין קשת ישירה בכיוון ההפוך, כלומר ממשבצת j למשבצת i, וגם, אין קשתות ישירות אחרים ממשבצת i למשבצת j. קוביליו חשב שהוא מוכן עם הלוח החדש, אבל לפתע הוא שם לב שהמצב שהוא רוצה (כשאתה עובר ממשבצת למשבצת, באמצעות הקשתות, אתה לא יכול לחזור למשבצת שביקרת בעבר) לא מתאים לרשימה של קשתות. הוא חשב תחילה להסיר חלק מהקשתות הישירות, אבל זה יביא לשכתוב הרשימה, שיכולה להיות ממש ארוכה. זו הסיבה שקוביליו החליט להפוך את הכיוון של חלק מהקשתות המכוונות

## המשימה

אתה משחק מונופול באופן קבוע עם קוביליו. לכן אתה רוצה לעזור לו על ידי כתיבת התוכנית למנופול. התוכנית צריכה לומר לו אילו קשרים להפוך כדי שהתנאי האמור יחזיק מעמד. על התוכנית להכיל את הפונקציה find\_reverse אשר תתלנקג׳ (תורכב) ביחד עם התוכנית של השופטים.

## פרטי מימוש

**הפונקציה find\_reverse צריכה להתאים להגדרה הבאה**

*std::string find\_reverse (int N, int M, int connections[][2]);*

*היא נקראת רק פעם אחת על ידי תוכנית השיפוט עם שלושה פרמטרים N מספר המשבצות בלוח החדש; M מספר הקשתות ברישמה של קובילו ורשימה של M קשתות כל קשת מוצגת בעזרת 2 מספרים שלמים ומייצגת קשת מהמספר הראשון למספר השני*

*אתה צריך להחזיק מחרוזת באורך M תווים כל תו הוא 1 אם צריך להפוך את הקשת ו0 אחרת*

*אם יש מספר פתרונות אתה יכול להחזיר כל פתרון*

אתם נדרשים להגיש את הקובץ Monopoly.cpp שמכיל את הפונקציה find\_reverse

התוכנית יכולה להכיל פונקציות וקוד נוסף אבל אסור לה להכיל Mainבנוסף אסור לכם לקרוא מהקלט הסטנדרטי או להוציא פלט

## אילוצים

* 3 ≤ ***N*** ≤ 5 × 105
* 3 ≤ ***M*** ≤ 1.5 × 106

## תת משימות

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **משימה** | **נקודות** | ***N*** | ***M*** | **מגבלות נוספות** |
| 1 | 0 | – | – | הדוגמא |
| 2 | 15 | ≤ 7 | ≤ 21 | – |
| 3 | 40 | *≤* 103 | ≤ 5 × 103 | – |
| 4 | 25 | ≤ 105 | ≤ 5 × 105 | – |
| 5 | 20 | ≤ 5 × 105 | ≤ 1.5 × 106 | – |

*כדאי לקבל ניקוד חלקי על תת משימה צריך לעבור את כל הבדיקות של התת משימה*

## בדיקה מקומית

**לצורך בדיקה מקומית על המחשב שלכם אתם מקבלים את התוכנית**

**Lgrader.cpp**

**אתם צריכים לשים אותה באותה ספריה כמו התוכנית**

**Monopoly.cpp**

**ואתם צריכים לקמפל את שתי התוכניות ביחד**

**התוכנית מקבלת מהקלט הסטדנרטי את הפרמטרים הבאים**

**בשורה הראשונה שני מספרים**

**N מספר המשבצות וM מספר הקשתות**

**בM השורות הבאות יתקבלו Mקשתות**

**כל שורה תכלול 2 מספרי קודקודים**

**כאשר כל שורה תייצג קלט בודדת בין המספר הראשון שנקלט בא למספר השני**

**הפלט של התוכנית זה המחרוזת הבינארית שאתם מחזירים**

## דוגמא לבדיקות מקומיות

|  |  |  |
| --- | --- | --- |
| **קלט** | **פלט** | **הסבר** |
| 6 12 1 2 1 5 2 3 2 5 3 4 3 6 4 1 4 5 5 3 5 6 6 1 6 4 | 100000101011 | הקשתות הבאות יחליפו כיוון:  (1,2), (4, 1), (5, 3), (6, 1), (6, 4).    שתי הדיאגרמות מדגימות את המשבצות ראשית המקורית והשניה אחרי היפוכים  בהתחלה אנחנו רואים שהקשתות המכוונות  (1, 2), (2, 3), (3, 4) , (4, 1)  מאפשרות חזרה לאותה משבצת שכבר ביקרנו בה  . אפשר לראות שהדרישה של קוביליו מחזיקה בלוח הסופי  יש פתרונות נוספים  (4, 1), (5, 3), (6, 1), (6, 4)  (1, 5), (2, 3), (2, 5), (3, 4), (3, 6), (4, 5), (5, 6) |