

السؤال 1 (C# / Java) من بجروت علم الحاسوب — صيف 2024، النموذج 899271

(في الامتحان فصلان. يجيب الطالب عن أربعة أسئلة من الفصلين، ولكل سؤال 25 علامة. أسئلة الفصل الأول (من 1 حتى 5) تُكتب بلغة Java أو بلغة C#، والطالب يكتب بلغة واحدة فقط. هنا نكتب الحل باللغتين: Java أولاً ثم C#.)
ملاحظة: في كل سؤال يُطلب فيه استقبال، لا حاجة لفحص سلامة المدخلات.
للذين يحلون بلغة Java: في كل سؤال يُطلب فيه استقبال، افترضوا أن الأمر التالي مكتوب في البرنامج:

Scanner input = new Scanner(System.in);

انتبهوا: في كل سؤال يُطلب فيه تطبيق، بإمكانكم استعمال عمليات الفئات:
دور وراصة وشجرة بينارية وحلقة، بدون تطبيقها. إذا استعملتم عمليات إضافية، يجب تطبيقها.

الفصل الأول

1. "حدّ سحري" هو حدّ في دور أعداد، قيمته تساوي مجموع قيمتي الحد الذي قبله والحد الذي بعده.
ملاحظة: العدد الأول في الدور والعدد الأخير في الدور ليسا "حدّين سحريين".
أ. اكتبوا عملية باسم isMagic بلغة Java أو IsMagic بلغة C#، تتلقّى دوراً q من نمط صحيح، وعدداً صحيحاً m أكبر من 0 وأصغر أو مساوٍ لأكبر الدور.
تعيد العملية true إذا كان الحد في المكان m في الدور هو "حدّ سحري"، خلاف ذلك، تُعيد العملية false.
ملاحظتان: — في نهاية العملية، يجب الحفاظ على مبنى الدور كما تمّ تلقّيه.
— لا تستعملوا في هذا البند مصفوفة أو قائمة مرتبطة. الحل الذي يتضمن استعمالهما، لن يحصل على درجات.

مثال: بالنسبة للدور الذي أمامكم:

رأس الدور	1	2	3	4	5	6	7	نهاية الدور
	5	11	6	9	3	6	3	

- بالنسبة لـ $m = 1$ ، تُعيد العملية false (العدد الأول في الدور ليس "حدّاً سحرياً").
بالنسبة لـ $m = 2$ ، تُعيد العملية true ($5 + 6 = 11$).
بالنسبة لـ $m = 3$ ، تُعيد العملية false ($11 + 9 \neq 6$).

ب. اكتبوا عملية باسم nMagic بلغة Java أو NMagic بلغة C# ، تتلقى دوراً من نمط صحيح – q ، وعدداً صحيحاً n أكبر من 0 وأصغر أو مساوٍ لِكِبَر الدَّور .
تُعيد العملية true إذا كانت جميع الحدود الواقعة في الأماكن التي هي مضاعفات لـ n (المكان الـ n في الدَّور والمكان الـ 2n في الدَّور وهكذا، بتخطيات n أماكن) هي "حدود سحرية" . خلاف ذلك ، تُعيد العملية false .
يمكن استعمال العملية التي كتبتموها في البند "أ" .
ملاحظتان: – في هذه العملية ، لا حاجة للحفاظ على الدَّور الذي تم تلقيه .
– لا تستعملوا في هذا البند مصفوفة أو قائمة مرتبطة . الحل الذي يتضمن استعمالهما ، لن يحصل على درجات .

أمثلة : بالنسبة للدَّور الذي في صفحة 2 :
بالنسبة لـ $n = 2$ ، تُعيد العملية true ، لأن جميع الحدود الواقعة في الأماكن التي هي مضاعفات لـ 2 (2 ، 4 ، 6) هي "حدود سحرية" .
بالنسبة لـ $n = 4$ ، تُعيد العملية true ، لأن الحدّ الواقع في المكان 4 هو "حدّ سحري" (لا توجد في الدَّور حدود أخرى في الأماكن التي هي مضاعفات لـ 4) .
بالنسبة لـ $n = 3$ ، تُعيد العملية false ، لأن الحدّ في المكان 3 ليس "حدّاً سحريّاً" .

الحل: أ. نمزّ على الدور وننسخه إلى دور مساعد، ونحفظ الحدّ في المكان m مع جاريه
ب. مرور واحد: نفحص كلّ مكان من مضاعفات n

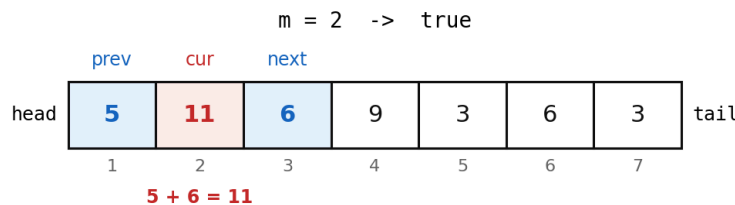
(الحلّ والشرح: فيفو · vevo . لا تنشر وزارة التربية حلولاً لهذا الامتحان. لا يوجد مترجم Java ولا C# على جهازنا، لذلك ترجمنا الكود سطرًا سطرًا إلى Python وشغلناه فعليًا على مثال الورقة، وعلى نحو 150 ألف حالة عشوائية قارناها بتعريف «الحدّ السحري» مباشرة.)

الحلّ الكامل:

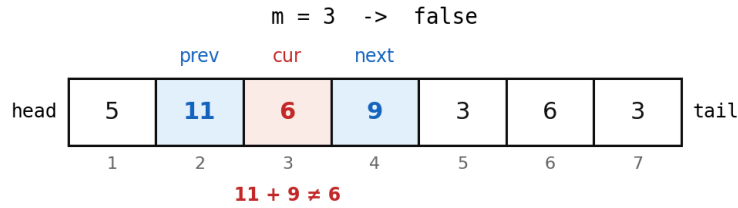
البند أ

الفكرة: الحدّ في المكان m سحريّ إذا كانت قيمته تساوي مجموع الحدّ الذي قبله (المكان $m-1$) والحدّ الذي بعده (المكان $m+1$). لذلك نحتاج ثلاث قيم فقط: prev, cur, next. ممنوع استعمال منظومة أو قائمة، فنحفظ القيم الثلاث في متغيّرات عادية.

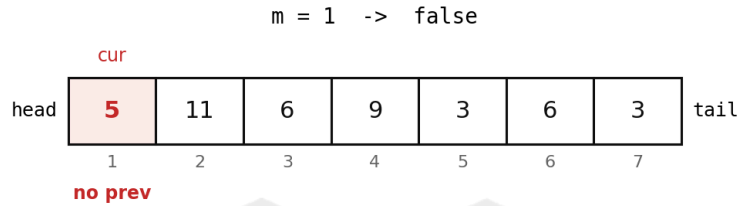
مثال الورقة مع $m = 2$: الحدّ 11 بين الحدّين 5 ثمّ 6، والمجموع $5 + 6 = 11$ ، فالعملية تعيد true:



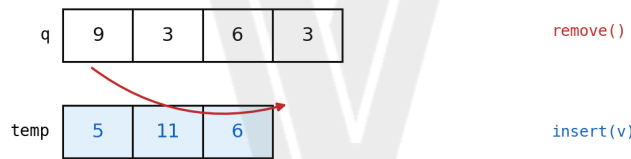
مع $m = 3$: الحد 6 بين الحدين 11 ثم 9، والمجموع $11 + 9 \neq 6$ ، فالعملية تعيد false:



مع $m = 1$: لا يوجد حدّ قبله، فهو ليس سحريًا أبدًا. وكذلك الحدّ الأخير (لا يوجد حدّ بعده). لهذا نفحص في الكود أنّ $m > 1$ وأنّ next موجود فعلاً:



يجب أن يبقى الدور كما تلقيناه. الدور يسمح بالوصول إلى الرأس فقط، لذلك نخرج كلّ الحدود بالترتيب وندخلها إلى دور مساعد temp، ثمّ نعيدها كلّها إلى q بالترتيب نفسه:



منتصف المرور: أخرجنا ثلاثة حدود من q وأدخلناها إلى temp.

الكود بلغة Java. السطر 8 حتّى السطر 10 تحفظ الحدود الثلاثة، والسطر 14 يعيد الدور، والسطر 15 هو الشرط النهائي:

Java

```
1 public static boolean isMagic(Queue<Integer> q, int m) {
2     Queue<Integer> temp = new Queue<Integer>();
3     int pos = 0, prev = 0, cur = 0, next = 0;
4     boolean hasNext = false;
5     while (!q.isEmpty()) {
6         int v = q.remove();
7         pos++;
8         if (pos == m - 1) prev = v;
9         else if (pos == m) cur = v;
10        else if (pos == m + 1) { next = v; hasNext = true; }
11        temp.insert(v); // keep a copy
12    }
13    while (!temp.isEmpty()) // restore q
14        q.insert(temp.remove());
15    return m > 1 && hasNext && cur == prev + next;
16 }
```

الكود بلغة C# (المنطق نفسه؛ هنا السطر 10 حتى السطر 12 تحفظ الحدود، والسطر 17 هو الشرط):

C#

```
1 public static bool IsMagic(Queue<int> q, int m)
2 {
3     Queue<int> temp = new Queue<int>();
4     int pos = 0, prev = 0, cur = 0, next = 0;
5     bool hasNext = false;
6     while (!q.IsEmpty())
7     {
8         int v = q.Remove();
9         pos++;
10        if (pos == m - 1) prev = v;
11        else if (pos == m) cur = v;
12        else if (pos == m + 1) { next = v; hasNext = true; }
13        temp.Insert(v); // keep a copy
14    }
15    while (!temp.IsEmpty()) // restore q
16        q.Insert(temp.Remove());
17    return m > 1 && hasNext && cur == prev + next;
18 }
```

(نتيجة التشغيل على مثال الورقة: الأماكن 2, 4, 6 تعطي true، والأماكن 1, 3, 5, 7 تعطي false. وفحصنا بعد كل استدعاء أن الدور بقي كما كان.)

return m > 1 && hasNext && cur == prev + next;

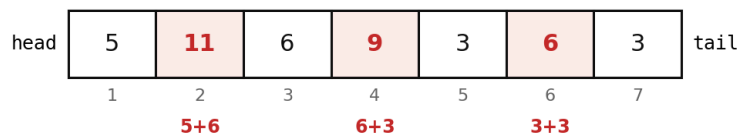
الجواب النهائي للبند أ (قلب الحل)

البند ب

الأماكن المطلوبة هي n , $2n$, $3n$ ، وهكذا. يمكن أن نستدعي isMagic لكل مكان منها، لكن كل استدعاء يمر على الدور كله. الأسرع: مرور واحد على الدور. هنا لا يجب الحفاظ على الدور، فنخرج الحدود واحدًا واحدًا ونحفظ آخر ثلاثة.

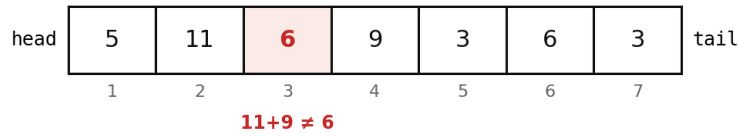
مثال الورقة مع $n = 2$: الأماكن 2, 4, 6 كلها سحرية، فالجواب true:

n = 2 : positions 2, 4, 6 -> true



مع $n = 3$: المكان 3 ليس سحريًا، فنعيد false فورًا ولا نكمل:

$n = 3$: position 3 fails -> false



في كل دورة: cur هو الحد في المكان pos، والمتغير next هو الحد بعده. إذا كان pos من مضاعفات n، نفحص أنه سحري (المكان 1 ليس سحريًا أبدًا). بعد انتهاء الدور يبقى الحد الأخير؛ إذا كان مكانه من مضاعفات n فالجواب false، لأن الحد الأخير ليس سحريًا.

الكود بلغة Java. السطر 6 هو الفحص، والسطر 12 يعالج الحد الأخير:

Java

```
1 public static boolean nMagic(Queue<Integer> q, int n) {
2     int pos = 1, prev = 0;
3     int cur = q.remove();           // position 1
4     while (!q.isEmpty()) {
5         int next = q.remove();
6         if (pos % n == 0 && (pos == 1 || cur != prev + next))
7             return false;
8         prev = cur;
9         cur = next;
10        pos++;
11    }
12    return pos % n != 0;             // last element is never magic
13 }
```

الكود بلغة C# (السطر 8 هو الفحص، والسطر 14 يعالج الحد الأخير):

C#

```
1 public static bool NMagic(Queue<int> q, int n)
2 {
3     int pos = 1, prev = 0;
4     int cur = q.Remove();           // position 1
5     while (!q.IsEmpty())
6     {
7         int next = q.Remove();
8         if (pos % n == 0 && (pos == 1 || cur != prev + next))
9             return false;
10        prev = cur;
11        cur = next;
12        pos++;
13    }
14    return pos % n != 0;             // last element is never magic
15 }
```

n	الأماكن المفحوصة	النتيجة
2	2, 4, 6	true
4	4	true
3	3	false
1	1	false
7	المكان 7 (الأخير)	false

(حلّ بديل مقبول: نحسب طول الدور، ثمّ نستدعي isMagic للأماكن $n, 2n, 3n, \dots$ حتّى الطول. هو صحيح لكنّه أبطأ: تعقيده تربيعيّ في طول الدور، أمّا حلّنا فخظيّ.)

```
if (pos % n == 0 && (pos == 1 || cur != prev + next)) return false;
```

الجواب النهائيّ للبند ب (قلب الحلّ)

نصّ الأسئلة والصور: وزارة التربية والتعليم، النموذج 899271، صيف 2024. الحلّ والشرح: فيفو · vevo (لا تنشر وزارة التربية حلولاً لهذا الامتحان). حقوق الطبع للوزارة: «النسخ أو النشر ممنوعان إلا بإذن من وزارة التربية والتعليم».

