Çakıl Taşlı Mesafe Ölçer

Leonardo orijinal 'Mesafeölçer' i (tekerlekleri döndükçe yere çakıltaşları bırakarak mesafe ölçebilen bir araç) icat etmiştir. Yere düşen çakıltaşlarının sayısı tekerleğin kaç kez döndüğünün bulunmasını sağlar, böylelikle kullanıcı mesafeölçerin kat ettiği mesafeyi de hesaplayabilir. Biz de bilgisayar bilimcileri olarak mesafeölçerin fonksiyonalitesini artıracak yazılım kontrolü ekledik. Sizin göreviniz, aşağıdaki belirtilen kurallar altında mesafeölçeri programlamak.

Operasyon Izgarası

Mesafeölçer, 256 × 256 boyutunda hayali bir karesel ızgara üzerinde hareket eder. Her bir hücre en fazla 15 çakıltaşı alabilmektedir ve (satır, sütun) koordinatları ile anılır. Satır ve sütunlar 0 dan 255 e kadar değerler alabilir. Bir (i, j) hücresi verildiğinde buna komşu hücreler (eğer varsa) (i - 1, j), (i + 1, j), (i, j - 1) ve (i, j + 1) dir. İlk ya da son satır, veya ilk ya da son sütunda, yer alan hücreler *sınır* hücresi olarak adlandırılır. Mesafeölçer her zaman (0, 0) (kuzey-batı köşesi) hücresinden yüzü kuzeye dönük olarak başlar.

Basit komutlar

Mesafeölçer aşağıdaki komutlar kullanılarak programlanabilir.

- `left` — 90 derece sola dön (saat yönünün tersi istikamette) ve bulunduğun hücrede kal (mesela eğer öncesinde yüzü güneye dönükse, komuttan sonra yüzü doğuya dönük olur).
- `right` — 90 derece sağa dön (saat yönü istikametinde) ve bulunduğun hücrede kal (mesela eğer öncesinde yüzü batıya dönükse, komuttan sonra yüzü kuzeye dönük olur).
- `move` — (yüzünün dönük olduğu yönde) bir hücre ileriye git. Eğer gidilecek hücre yoksa (mesela zaten o doğrultudaki sınıra ulaşılmışsa) bu komut etkisizdir.
- `get` — o anki hücreden bir çakıltaşını kaldır. Eğer hücrede çakıltaşı yoks, bu komut etkisizdir.
- `put` — o anki hücreye bir çakıltaşı ekle. Eğer o anki hücrede zaten 15 çakıltaşı varsa, bu komut etkisizdir. Mesafeölçerde çakıltaşı hiç bitmemektedir.
- `halt` — programı durdur.

Mesafeölçer komutları programda (her satırda bir komut olacak şekilde) verilen sırada çalıştırır. Boş satırlar dikkate alınmaz. # yorum belirtir, ve bu sembolden başlayıp, o satırın sonuna kadar olan bütün karakterler görmezden gelinir. Mesafeölçer programın sonuna ulaşırsa, programı çalıştırmayı durdurur.

Örnek 1

Mesafeölçer için yazılmış aşağıdaki programı göz önünde bulundurunuz. Bu program, mesafeölçeri (0,2) koordinatlı hücreye, yüzü doğuya dönük olacak şekilde götürür. (İlk `move` komutunun görmezden gelindiğine dikkat edin. Mesafeölçer kuzey-batı köşesinde yüzü kuzeye dönük olduğundan bu komut uygulanmamıştır.)

```
move  # etki yok
right
# Şu anda mesafeölçer doğuya bakmaktadır.
move
move
```

Etiketler, sınırlar ve çakıltaşları

O anki duruma bağlı olarak program akışını değiştirmek için etiketler kullanabilirsiniz. Etiketler büyük-küçük harfe duyarlı, en fazla 128 karakter uzunluğunda, sadece `a`, …, `z`, `A`, …, `Z`, `0`, …, `9` karakterlerini içeren dizgilerdir. Etiketlerle alakalı yeni komutlar aşağıda listelenmiştir. Açıklamalarda, *L* herhangi bir etiketi temsil etmektedir.

- *L*`:` (*L* ve onu takip eden bir ikinokta üst üste '`:`') — programda *L* ile etiketlenmiş bir konumu belirtir. Belirtilmiş bütün etiketler birbirinden farklı olmalıdır. Etiket belirlemek mesafeölçer üzerinde bir etki yaratmaz.

- `jump` *L* — programın işleyişine, koşulsuz olarak *L* ile etiketlenmiş satıra giderek devam et.

- `border` *L* — eğer mesafeölçer bir sınırda ve yüzü ızgaranın dışına dönük ise (yani bir `move` komutu etkisiz olacaksa) *L* ile etiketlenmiş satıra giderek programa devam et. Eğer mesafeölçer bu durumda değilse, bu komutu yoksayarak programın işleyişine devam et.

- `pebble` *L* — eğer mesafeölçerin bulunduğu hücrede en az bir çakıltaşı varsa *L* ile etiketlenmiş satıra giderek programa devam et. Eğer mesafeölçer bu durumda değilse, bu komutu yoksayarak programın işleyişine devam et.

Örnek 2

Aşağıdaki program 0 numaralı satırdaki ilk (en batıdaki) çakıltaşını bulup orada durmaktadır. Eğer 0 numaralı satırda çakıltaşı yoksa, satırın sonundaki sınırda durmaktadır. Bu program, `leonardo` ve `davinci` etiketlerini kullanmaktadır.

```
right
leonardo:
pebble davinci  # çakıltaşı bulundu
border davinci  # satırın sonu
move
jump leonardo
davinci:
halt
```

Mesafeölçer sağa dönerek hareketine başlar. Döngü, `leonardo:` etiketinin tanımlanmasıyla başlayıp `jump leonardo` komutuyla biter. Döngünün içinde mesafeölçer bir çakıltaşının varlığını ya da satırın sonundaki sınırı arar. Eğer bulamazsa, o an bulunduğu (0,j) hücresinden

(0,j+1) hücresine `move` komutuyla geçer, (0,j+1) hücresi var olduğundan. (burada `halt` komutu tamamen gerekli değildir, ne de olsa program her türlü sonlandırılmaktadır.)

Görev İfadesi

Yukarıda tanımlandığı biçimde, mesafeölçerin istenildiği gibi davranmasını sağlayacak kendi dilinde bir program göndermelisiniz. Her altgörev (aşağıda tanımlandığı gibi) gönderilen çözümün sağlaması gereken kısıtları ve mesafeölçerin gerçekleştirmesi gereken davranışları belirtmektedir. Çözümün kısıtlarında iki nokta göz önünde bulundurulmaktadır.

- *Program boyutu* — yazılan program yeterince kısa olmalı. Bir programın boyu, içerdiği komut sayısıdır. Etiket tanımlamaları, komutlar ve boş satırlar program boyutunda *sayılmamaktadır*.

- *Çalıştırma süresi* — program yeterince çabuk çalışmalıdır. Çalıştırma süresi uygulanan *işlem* sayısıdır. Çalıştırılmış her bir komut (etkisiz olsa dahi) bir işlem sayılmaktadır. Etiket tanımlamaları, yorumlar ve boş satırlar işlem sayılmamaktadır.

Birinci örnekte, program boyutu 4 ve çalıştırma süresi 4 tür. İkinci örnekte, program boyutu 6 dır, ve içinde sadece (0,10) hücresinde bir tane çakıltaşı bulunan bir ızgarada çalıştırıldığında çalıştırma süresi 43 işlemdir. Bu 43 işlem sırasıyla; `right`, döngü için 10 iterasyon, her iterasyonda 4 işlem (`pebble davinci`; `border davinci`; `move`; `jump leonardo`) ,`pebble davinci` ve son olarak `halt` dır.

Altgörev 1 [9 puan]

Başlangıçta (0,0) hücresinde x, (0,1) hücresinde y adet çakıltaşı vardır, ve diğer bütün hücreler boştur. Unutmayınız ki her hücrede en fazla 15 çakıltaşı bulunabilir. Mesafeölçeri, eğer $x \leq y$ ise (0,0) da, değilse (0,1) de sonlandıran bir program yazınız. (Programın sonunda mesafeölçerin yüzünün döndüğü yön ve diğer hücrelerde kaç adet çakıltaşı olduğu önemli değildir).

Sınırlar: program boyutu ≤ 100, çalıştırma süresi ≤ 1 000.

Altgörev 2 [12 puan]

Yukarıdaki görev ile aynı fakat program sonlandığında (0,0) hücresinde x, (0,1) hücresinde y adet çakıltaşı bulunmalıdır.

Sınırlar: program boyutu ≤ 200, çalıştırma süresi ≤ 2000.

Altgörev 3 [19 puan]

0 numaralı satırda tam olarak iki tane çakıltaşı var; biri (0,x), öbürü ise (0,y) hücresinde. x ile y birbirinden varklı, ve x+y çift. Mesafeölçeri (0,(x+y)/2) hücresinde, yani çakıltaşı içeren iki hücrenin tam ortasındaki hücrede bırakan bir program yazınız. Izgaranın son durumu önemli değil.

Sınırlar: program boyutu ≤ 100, çalıstırma süresi ≤ 200 000.

Altgörev 4 [32 puana kadar]

Izgarada en fazla 15 tane çakıltaşı var, ve herhangi bir hücrede birden fazla çakıltaşı yok. Bütün çakıltaşlarını kuzey-batı köşede toplayan bir program yazınız. Daha açık olarak, başlangıçta ızgarada x tane çakıltaşı varsa, sonunda (0,0) hücresinde tam olarak x tane çakıltaşı olmalı, ve başka hiçbir yerde çakıltaşı olmamalı.

Bu altgörev için puan, çalıştırma süresine bağlıdır. Daha net bir şekilde, L, farklı girdiler için en uzun çalıştırma süresi ise, sizin skorunuz;

- $L \leq 200\,000$ ise 32 puan;
- $200\,000 < L < 2\,000\,000$ ise $32 - 32 \log_{10} (L / 200\,000)$ puan;
- $L \geq 2\,000\,000$ ise 0 puan.

Sınırlar: program boyutu ≤ 200.

Altgörev 5 [28 puana kadar]

Izgaranın herhangi bir hücresinde herhangi bir sayıda çakıltaşı olabilir (tabii ki 0 ile 15 arasında). Bu sayıların minimumunu bulan bir program yazınız; yani program sonlandığında mesafeölçerin bulunduğu (i,j) hücresi,diğer bütün hücrelerin en az (i,j) hücresinde bulunan çakıltaşı miktarı kadar çakıltaşı içermesini garanti edecek bir hücre olmalıdır. Program çalıştırıldıktan sonra, bütün hücreler programın başında ne kadar çakıltaşına sahipse o kadar çakıltaşı içermelidir.

Bu altgörev için puan, program boyutuna bağlıdır. Daha net bir şekilde, P, yazdığınız programın boyutu ise, sizin skorunuz;

- $P \leq 444$ ise 28 puan;
- $444 < P < 4\,440$ ise $28 - 28 \log_{10} (P / 444)$ puan;
- $P \geq 4\,440$ ise 0 puandır.

Sinirlar: çalistirma süresi ≤ 44 400 000.

Gerçekleştirme detayları

Her altgörev için yukarıdaki program kurallarına uyan tam olarak bir tane gönderim yapmalısınız. Gönderilerin dosyaların boyutu en fazla 5 MiB olmalıdır. Her bir altgörev için, mesafeölçer kodunuz birkaç girdi üzerinde denenecektir, ve kullanmış olduğunuz süre ve hafıza hakkında bilgilendirileceksiniz. Kodunuzun sözdiziminin hatalı olması durumunda, test etmek imkansız

olduğundan, ilgili sözdizimi hatası hakkında bilgi alacaksınız.

Gönderiminizin bütün altgörevler için mesafeölçer programları barındırması gerekmemektedir. Eğer son gönderiminizde altgörev X için bir program yoksa, altgörev X için geçerli son programınız otomatik olarak eklenmektedir. Eğer böyle bir program yoksa, altgörev X için puanınız 0 olacaktır.

Her zamanki gibi, bir gönderimin puanı bütün altgörevlerden alınan puanın toplamına eşittir, ve bu soru için alacağınız puan şu ana kadar test ettiğiniz ve en son gönderdiğiniz kodlar arasından en yüksek puanlısıdır.

Simulatör

Programınızı test edebilmeniz için, size girdi dosyası ve mesafeölçer programı sağlayabileceğiniz bir mesafeölçer simülatörü verilmiştir. Test programı için verilecek mesafeölçer programları gönderimleriniz ile aynı formattadır (yani soruda belirtildiği gibi).

Izgara, programa bu formatta verilecektir; girdinin her satırında bir hücrenin satır, sütün ve içindeki çakıltaşı sayısını belirten R, C ve P sayıları yer alacaktır. Girdide belirtilmemiş diğer bütün hücreler başlangıçta 0 adet çakıltaşı içeriyor varsayılacaktır. Örnek olarak bu dosyaya bakınız:

```
0 10 3
4 5 12
```

Bu dosya ile tanımlanmış ızgarada 15 çakıltaşı mevcuttur; 3 tanesi (0,10) hücresinde, ve 12 tanesi (4,5) hücresinde.

Görev dizinindeki `simulator.py` programını, yazdığınız mesafeölçer programının ismini argüman olarak vererek çalıştırdığınızda test simülatörünü başlatırsınız. Simülatör programı aşağıdaki komut satırı seçeneklerini kabul edecektir:

- `-h` kullanabileceğiniz seçeneklerin kabaca tanımını yapar;
- `-g GRID_FILE` `GRID_FILE` dosyasından ızgarayı okur. (varsayılan: boş ızgara);
- `-s GRID_SIDE` ızgaranın boyunu `GRID_SIDE x GRID_SIDE` olarak değiştirir. (varsayılan: 256, soruda tanımladığı gibi). Daha küçük ızgara boyutları debug etmenizi kolaylaştırabilir.
- `-m STEPS` programın çalıştırma süresini en fazla `STEPS` işlem olmak üzere kısıtlar.
- `-c` derleme moduna girer. Derleme modunda, simülatör aynı çıktıyı verir, fakat Python ile simüle etmek yerine, küçük bir C programı oluşturup derler. Bu başlangıçta çok fazla işlem yapılmasına sebep olur, fakat zaman geçtikçe çok daha hızlı sonuçlar verir. Programınızın 10 000 000 civarında işlem yapmasını bekliyorsanız bu modu kullanmanız önerilir.

Gönderim sayısı

Bu görev için maksimum gönderim sayısı 128 dir.