Son Akşam Yemeği

Leonardo, en ünlü freski olan Son Akşam Yemeği üzerinde uğraşırken çok aktif bir şekilde çalışıyordu: Bir gün içinde yaptığı işlerden ilki günün geri kalanında kullanacağı boya renklerini seçmekti. Gün içinde bir sürü renge ihtiyacı oluyordu fakat sadece belirli bir sayıdaki renkleri iskelesinde barındırabiliyordu. Asistanı, başka işlerinin arasında, iskeleye tırmanıp Leonardo'nun istediği renkleri ona götürmekten ve iskeledeki renkleri alıp geri ait oldukları yere koymaktan sorumluydu.

Bu görevde, asistana yardım etmek için iki farklı program yazmanız gerekmektedir. İlk program Leonardo'nun talimatlarını (Leonardo'nun gün içinde ihtiyacı olacak renklerin dizisi) alacak, ve asistan için *tavsiye* isimli *kısa* bir dizgi yaratacaktır. Gün içinde, asistan Leonardo'nun isteklerine erişemeyecek, sadece ilk program tarafından oluşturulmuş tavsiye listesine erişimi olacak. İkinci program ise tavsiyeyi alacak, ve Leonardo'nun isteklerine çevrimiçi bir şekilde (yani her seferde bir tane olmak üzere) erişip onları işleyecek. Bu programın tavsiyenin ne anlama geldiğini anlayabilmesi ve optimal seçimleri yapabilmesi gerekmektedir. Her şey aşağıda daha detaylı bir şekilde anlatılmaktadır.

Renkleri raflar ve iskele arasında hareket ettirmek

Basitleştirilmiş bir senaryoyu göz önünde bulunduracağız. 0'dan N-1'e kadar numaralandırılmış N adet renk olduğunu, ve de Leonardo'nun her gün asistanından tam olarak N kere renk isteyeceğini varsayalım. C dizisi, Leonardo'nun asistanından istediği N rengi içeren dizi olsun. Böylece, C'yi her elemanı 0 ile N-1 (dahil) arasından seçilmiş N elemanlı bir tamsayı dizisi olarak düşünebiliriz. Bazı renklerin C'de hiç bulunmayabileceğini, ve bazılarının da C'de birden fazla kere bulunabileceğini göz önünde bulundurun.

İskele her zaman doludur ve N renkten K tanesini barındırmaktadır (K<N). Başlangıçta, iskele 0 ile K-1 (dahil) arasındaki renkleri barındırmaktadır.

Asistanı Leonardo'nun isteklerini birer birer işlemektedir. Eğer istenilen renk *zaten iskelenin üzerindeyse*, asistan dinlenebilir. Durum bu değilse, istenilen rengi raftan alıp iskeleye koymak zorundadır. Tabii ki iskelede yeni renk için yer olmadığından, asistan iskeledeki renklerden birini de seçip rafına geri koymak zorundadır.

Leonardo'nun optimal stratejisi

Asistan olabildiğince çok dinlenmek istemektedir. Dinlenebileceği isteklerin sayısı bu süreçteki seçimlerine bağlıdır. Daha net olmak gerekirse, asistanın iskeleden bir renk almasının gerektiği her seferde, farklı seçimler gelecekte farklı sonuçlara yol açabilir. Leonardo, C'yi bildiği takdirde bu amacına nasıl ulaşabileceğini asistanına açıklar. Hangi rengin kaldırılacağına karar vermek için

yapılacak en iyi seçim iskelede o an bulunan renkleri, ve C dizisinde geri kalan renk isteklerini gözlemleyerek elde edilebilir. İskelede bulunan renklerden kaldırılacak olan aşağıdaki kurallara göre belirlenmelidir:

- Eğer iskelede daha sonra gerekmeyecek bir renk varsa, asistan bu rengi iskeleden kaldırmalıdır.
- Eğer böyle bir renk yoksa, iskeleden kaldırılacak renk *gelecekte en geç ihtiyaç duyulacak renk* olmalıdır. (Yani, iskeledeki bütün renkler için o rengin gelecekte ilk ihtiyaç duyulduğu anı buluyoruz. Rafa geri kaldırılan renk en geç ihtiyaç duyulan renk olacaktır.)

Leonardo'nun stratejisini kullanarak, asistanın olabilecek en fazla kez dinlenebileceği kanıtlanabilir.

Örnek 1

N = 4 olsun; yani 4 tane rengimiz (0'dan 3'e kadar numaralandırılmış) ve 4 tane isteğimiz olsun. İstek dizisi de C = (2, 0, 3, 0) olsun. Bunun dışında, K = 2 olduğunu varsayın. Yani, Leonardo aynı anda 2 tane renk barındırabilecek bir iskeleye sahip. Yukarıda belirtildiği üzere, iskele başlangıçta 0 ile 1 renklerini barındırıyor. İskelenin içeriğini şu şekilde yazacağız: [0, 1]. Asistanın istekleri işleyebileceği bir senaryo aşağıdadır.

- İlk istenilen renk (2 numara) iskelede değil. Asistan bu rengi iskeleye koyar ve 1 numaralı rengi iskeleden kaldırmaya karar verir. O anki iskelenin durumu [0, 2]'dir.
- Sıradaki istenilen renk (0 numara) zaten iskelede, yani asistan dinlenebilir.
- Üçüncü istek (3 numara) için asistan 0 numaralı rengi kaldırır, ve iskelenin durumunu [3, 2]'ye çevirir.
- Son olarak, en son istenilen renk (0 numara) raftan alınıp iskeleye konmak zorundadır. Asistan 2 numaralı rengi iskeleden almaya karar verir, ve iskelenin durumu [3, 0] olur.

Yukarıdaki örneğin Leonardo'nun optimal stratejisini izlemediğine dikkat edin. Optimal strateji üçüncü aşamada 2 numaralı rengi kaldırmayı gerektirirdi, böylelikle asistan son aşamada bir kez daha dinlenebilirdi.

Hafızası kısıtlıyken asistanın stratejisi

Sabahleyin, asistan Leonardo'dan C dizisini bir kağıda yazmasını ister ki optimal stratejiyi bulabilsin. Fakat, Leonardo kendi çalışma tekniklerini gizli tutmak konusunu kafaya taktığından, asistanın kağıdı almasına izin vermez. Onun yerine asistanın sadece C dizisini okuyup aklında tutmasına izin verir.

Maalesef, asistanın hafızası gerçekten kötü, ve sadece M bit hatırlayabilir. Bu durum, asistanın bütün C dizisini tekrardan oluşturmasını engelleyebilmektedir. Bu yüzden asistan zekice bir çözüm bulup, aklında tutabileceği bir dizi oluşturmalıdır. Biz bu diziye *tavsiye dizisi* diyeceğiz ve A ile göstereceğiz.

Örnek 2

Sabahleyin, asistan Leonardo'nun C dizisini içeren kağıdını alıp, bunu okuyup, gereken bütün

seçimleri yapabilir. Yapmayı seçeceği şeylerden biri de her bir istekten sonra iskelenin alacağı durumu incelemek olacaktır. Mesela, Örnek 1'de verilmiş (optimal olmayan) stratejiyi uygularken, iskelenin durumlarının dizisi [0, 2], [0, 2], [3, 2], [3, 0] olur. (Asistanın iskelenin başlangıç durumunun [0, 1] olduğunu bildiğini hatırlayın.)

Varsayalım ki M = 16, yani asistan 16 bite kadar bilgiyi hatırlayabiliyor. N = 4 olduğundan, her rengi 2 bit kullanarak saklayabiliriz. Bu sayede, 16 bit yukarıda belirtilmiş iskelenin durumlarını gösteren diziyi saklamaya yeterlidir. Bu şekilde asistan aşağıdaki tavsiye dizisini hesaplar: A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`).

Asistan, gün içerisinde bu tavsiye dizisini deşifre edip seçimlerini yapabilir.

(Tabii ki, M = 16 iken asistan bunun yerine C dizisinin tamamını hatırlamayı seçebilir, sadece bu 16 bitin 8ini kullanarak. Bu örnekte sadece iyi çözümlerden fedakarlık etmeden asistanın farklı seçenekleri olabileceğini göstermek istedik.)

Görev İfadesi

Aynı programlama dilinde *iki farklı program* yazmanız gerekmektedir. Bu programlar çalışma esnasında birbiriyle iletişim halinde bulunmasına izin verilmeden birbiri ardına çalıştırılacaktır.

İlk program sabah asistan tarafından kullanılan program olacaktır. Bu programa bir C dizisi verilecektir, ve A isimli bir tavsiye dizisi hesaplaması gerekmektedir.

İkinci program asistan tarafından gün içinde kullanılan program olacaktır. Bu program A isimli bir tavsiye dizisi alacaktır, ve sonrasında Leonardo'nun C dizisindeki istekleri işlemesi gerekmektedir. Dikkatinizi çekmek isteriz ki, C dizisi bu programa her seferinde bir istek olarak verilecektir, ve sıradaki isteği almadan önce programın bir önceki isteği işlemiş olması gerekmektedir.

Daha kesin olmak gerekirse, birinci programda `ComputeAdvice`(C, N, K, M) isimli fonksiyonu gerçekleştirmeniz gerekmektedir. Bu fonksiyona girdi olarak N elemanlı bir tamsayı dizisi C (her eleman 0, …, N - 1 arasında olmak üzere), iskeledeki renklerin sayısı K, ve de tavsiye için kullanabileceğiniz bit sayısı M verilecektir. Bu program M ya da daha az bit içeren A isimli bir tavsiye dizisi oluşturmalıdır. Sonrasında da bu program A dizisini her bit için sırasıyla aşağıdaki fonksiyonu çağırarak sisteme iletmelidir:

- `WriteAdvice`(B) — B bitini o anki tavsiye dizisi olan A'nın sonuna ekle. (Bu fonksiyonu en fazla M defa çağırabilirsiniz.)

İkinci programda `Assist`(A, N, K, R) fonksiyonunu gerçekleştirmelisiniz. Bu fonksiyonun girdisi tavsiye dizisi A, yukarıda tanımlandığı gibi N ve K tamsayıları, A dizisindeki bit sayısını belirten R sayısı (R ≤ M) olacaktır. Bu fonksiyon asistan için sizin önerdiğiniz stratejiyi, aşağıda size sağlanmış fonksiyonları kullanarak gerçekleştirmelidir:

- `GetRequest`() — Leonardo tarafından istenen sıradaki rengi döndürür. (Gelecekteki istekler hakkında bir bilgi verilmez.)

- `PutBack`(T) — T rengini iskeleden alıp rafına geri koy. Bu fonksiyonu sadece T iskelede o anda bulunan renklerden biriyse çağırabilirsiniz.

Çalıştırıldığında, `Assist` fonksiyonunuz `GetRequest` fonksiyonunu her seferinde Leonardo'nun sıradaki isteğini öğrenmek için tam olarak N kere çağırmalıdır. `GetRequest` fonksiyonuna yapılan her çağrıdan sonra, eğer döndürülen renk iskelede *değilse*, aynı zamanda kendi istediğiniz T değeriyle `PutBack`(T) fonksiyonunu *çağırmalısınız*. Eğer renk zaten iskelede ise, `PutBack` fonksiyonunu *çağırmamalısınız*. Bu koşulu sağlamamanız bir hata sayılmaktadır ve programınızın sonlandırılmasına neden olacaktır. Başlangıçta iskelede 0 ile K-1 (dahil) arasındaki renklerin bulunduğunu hatırlayınız.

Belirli bir test girdisi, eğer iki fonksiyonunuz da verilen bütün koşulları sağlarsa ve `PutBack` fonksiyonuna yapılan toplam çağrı sayısı Leonardo'nun optimal stratejisine *tam olarak* eşitse çözülmüş sayılacaktır. Eğer `PutBack` fonksiyonuna yapılan çağrı sayısını aynı sayıda tutan birden fazla strateji varsa, programınızın bunlardan herhangi birini kullanmasına izin verildiğine dikkat edin. (yani eğer ona eşdeğer iyilikte başka bir strateji varsa, Leonardo'nun stratejisini kullanmak zorunda değilsiniz.)

Örnek 3

Örnek 2'den devam edersek, `ComputeAdvice` fonksiyonunda A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`) dizisini hesapladığınızı varsayabiliriz. Sistemle iletişime geçebilmek için, sırasıyla şu çağırmaları yapmanız gerekir: `WriteAdvice`(`0`), `WriteAdvice`(`0`), `WriteAdvice`(`1`), `WriteAdvice`(`0`), `WriteAdvice`(`0`), `WriteAdvice`(`0`), `WriteAdvice`(`1`), `WriteAdvice`(`0`), `WriteAdvice`(`1`), `WriteAdvice`(`1`), `WriteAdvice`(`0`), `WriteAdvice`(`1`), `WriteAdvice`(`1`), `WriteAdvice`(`0`), `WriteAdvice`(`0`).

Bundan sonra ikinci fonksiyonunuz olan `Assist` çalıştırılır, ve yukarıdaki A dizisini, N=4, K=2 ve R=16 tamsayılarını girdi olarak alır. `Assist` fonksiyonu daha sonrasında `GetRequest` fonksiyonunu tam olarak N = 4 kere çağırmalıdır. Aynı zamanda, bu isteklerin bazılarından sonra, `Assist` fonksiyonunun uygun T değeriyle `PutBack`(T) fonksiyonunu çağırması gerekecektir.

Aşağıdaki tablo Örnek 1'deki (optimal olmayan) seçimlere denk gelen fonksiyon çağırma dizisini göstermektedir. Kısa çizgi `PutBack` fonksiyonunun çağırılmamasını temsil etmektedir.

`GetRequest()`	Aksiyon
2	`PutBack`(1)
0	-
3	`PutBack`(0)
0	`PutBack`(2)

Altgörev 1 [8 puan]

- $N \leq 5\,000$.
- En fazla M = 65 000 bit kullanabilirsiniz.

Altgörev 2 [9 puan]

- $N \leq 100\,000$.
- En fazla M = 2 000 000 bit kullanabilirsiniz.

Altgörev 3 [9 puan]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- En fazla M = 1 500 000 bit kullanabilirsiniz.

Altgörev 4 [35 puan]

- $N \leq 5\,000$.
- En fazla M = 10 000 bit kullanabilirsiniz.

Altgörev 5 [39 puana kadar]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- En fazla M = 1 800 000 bit kullanabilirsiniz.

Bu altgörev için alacağınız puan programınızın yolladığı tavsiye dizisinin uzunluğu olan R sayısıyla alakalıdır. Daha açık olmak gerekirse, eğer R_{max} (bütün test girdileri arasında) `ComputeAdvice` tarafından üretilen en uzun tavsiye dizisinin uzunluğuysa, puanınız:

- $R_{max} \leq 200\,000$ ise 39 puan;
- $200\,000 < R_{max} < 1\,800\,000$ ise $39 (1\,800\,000 - R_{max}) / 1\,600\,000$ puan;
- $R_{max} \geq 1\,800\,000$ ise 0 puan.

Gerçekleştirim detayları

Aynı programlama dilinde yazılmış tam olarak iki dosya yollamalısınız.

İlk dosyanın ismi `advisor.c`, `advisor.cpp` veya `advisor.pas` dır. Bu dosya `ComputeAdvice` fonksiyonunu yukarıda anlatıldığı gibi gerçekleştirmelidir, ve `WriteAdvice` fonksiyonunu çağırabilir. İkinci dosyanın ismi `assistant.c`, `assistant.cpp` veya `assistant.pas` dır. Bu dosya `Assist` fonksiyonunu yukarıda anlatıldığı gibi gerçekleştirmelidir ve `GetRequest` ile `PutBack` fonksiyonlarını çağırabilir.

Bütün fonksiyonlar için başlık yapıları aşağıdadır.

C/C++ programları

```
void ComputeAdvice(int *C, int N, int K, int M);
void WriteAdvice(unsigned char a);
```

```
void Assist(unsigned char *A, int N, int K, int R);
void PutBack(int T);
int GetRequest();
```

Pascal programları

```
procedure ComputeAdvice(var C : array of LongInt; N, K, M : LongInt);
procedure WriteAdvice(a : Byte);
```

```
procedure Assist(var A : array of Byte; N, K, R : LongInt);
procedure PutBack(T : LongInt);
function GetRequest : LongInt;
```

Belirtilen fonksiyonlar yukarıda anlatıldığı gibi çalışmalıdır. Bunlar haricinde kendiniz ilave yardımcı fonksiyonlarda yazabilirsiniz. C/C++ programları için, dahili fonksiyonlarınız `static` olarak beyan edilmelidir; örnek notlandırıcı bu iki fonksiyonu birbirine bağlayacağı için. Alternatif olarak, iki programda aynı isme sahip iki farklı fonksiyon yazmaktan da sakınabilirsiniz. Kodunuz hiçbir şekilde standart girdi/çıktı ve başka bir dosyadan okuma yazma yapmamalıdır.

Çözümünüzü kodlarken, aşağıdaki yönergelere uymak zorundasınız (yarışma ortamında bulacağınız şablonlar zaten aşağıda belirtilen gereksinimleri sağlamaktadır).

C/C++ programları

`advisor.h` dosyasını advisor kodunuzun başında, `assistant.h` dosyasını assistant kodunuzun başında include etmelisiniz. Bunu aşağıdaki satırları kaynak kodunuza ekleyerek elde edebilirsiniz:

```
#include "advisor.h"
```

veya

```
#include "assistant.h"
```

`advisor.h` ve `assistant.h` isimli iki dosya size yarışma ortamınızın içindeki dizinde sağlanmıştır ve aynı zamanda yarışma Ağ arayüzünde bulunmaktadır. Aynı kanallar üzerinden size kodunuzu derleyip çalıştırmanızı sağlayacak kod ve komut dosyaları sağlanmıştır. Açık olmak gerekirse, kodunuzu bu dosyaları içeren dizine taşıdıktan sonra, `compile_c.sh` veya `compile_cpp.sh` (kullandığınız dile göre) dosyalarından birini çalıştırmanız gerekmektedir.

Pascal programları

`advisorlib` ve `assistantlib` ünitelerini sırasıyla advisor ve assistant kodlarında kullanmanız gerekmektedir. Bunu aşağıdaki satırları kaynak kodunuza ekleyerek elde edebilirsiniz:

```
uses advisorlib;
```

ve

```
uses assistantlib;
```

`advisorlib.pas` ve `assistantlib.pas` isimli iki dosya size yarışma ortamınızın içindeki dizinde sağlanmıştır ve aynı zamanda yarışma Ağ arayüzünde bulunmaktadır. Aynı kanallar üzerinden size kodunuzu derleyip çalıştırmanızı sağlayacak kod ve komut dosyaları sağlanmıştır. Açık olmak gerekirse, kodunuzu bu dosyaları içeren dizine taşıdıktan sonra, `compile_pas.sh` dosyasını çalıştırmanız gerekmektedir.

Örnek Notlandırıcı

Örnek notlandırıcı girdiyi aşağıdaki formatta okumaktadır:

- satır 1: N, K, M;
- satırlar 2, …, N + 1: C[i].

Notlandırıcı önce `ComputeAdvice` fonksiyonunu çalıştıracaktır. Sonrasında, boşlukla birbirlerinden ayrılmış ve 2 ile sonlanan, tavsiye dizisinin bireysel bitlerini içeren `advice.txt` dosyasını oluşturacaktır.

Bundan sonra sizin `Assist` fonksiyonunuzu çalıştırıp, her satırında ya "`R [number]`" ya da "`P [number]`" formatında dizgiler bulunan bir çıktı üretecektir. İlk formattaki satırlar `GetRequest()` fonksiyonuna yapılan çağrıları ve alınan cevapları temsil etmektedir. İkinci formattaki satırlar `PutBack()` fonksiyonuna yapılan çağrıları ve geri koyulmasına karar verilen renkleri temsil etmektedir. Çıktı "`E`" satırıyla sonlandırılmaktadır.

Resmi notlandırıcının çalışma zamanı kodunuzun sizin bilgisayarınızdaki çalışma zamanından biraz farklı olabilir. Bu fark çok kayda değer olmayacaktır. Çözümünüzün zaman sınırı içinde çalışıp çalışmadığını görebilmek için test arayüzünü kullanabilirsiniz.

Süre ve Hafıza limiti

- Süre limiti: 7 saniye.
- Hafıza limiti: 256 MB.