D. Tahmin Etme Oyunu

Problem İsmi	Tahmin Etme Oyunu
Time Sınırı	4 saniye
Memory Sınırı	1 gigabyte

Lund'un eski bir kasabasında, 0 ile $N-1$ arasında indekslenmiş N tane evin sıralandığı bir sokak var. Emma bu evlerden birinde yaşar ve arkadaşları Anna ve Bertil bu evin hangisi olduğunu öğrenmek ister. Emma arkadaşlarına nerede yaşadığını söylemek yerine onlarla bir oyun oynamaya karar verir. Oyun başlamadan önce Anna ve Bertil sadece sokaktaki evlerin sayısını biliyor. Bu noktada, Anna ve Bertil pozitif bir K tamsayısını seçebilir ve bir strateji üzerinde anlaşabilirler. Fakat bundan sonra herhangi bir iletişim yasaktır.

Oyunun kendisi iki aşamadan oluşur. İlk aşamada Emma, ziyaret edilecek son ev kendi evi olacak şekilde evlerin sıralamasını seçer. Daha sonra, Anna'ya sırayı önceden söylemeden, bu sırayla tek tek evlere götürür. Emma'nın evi olmayan her ev için, Anna'nın evin ön kapısına bir tebeşirle 1 ile K arasında tek bir tamsayı yazmasına izin verilir. Ziyaret ettikleri son ev olan Emma'nın evi için, Emma kapıya kendisi 1 ile K arasında bir tamsayı yazar.

Oyunun ikinci aşamasında Bertil, 0 numaralı evden $N-1$ evine giden cadde boyunca yürür ve Emma'nın kapılara yazdığı tüm numaraları okur. Bertil, Emma'nın hangi evde yaşadığını tahmin etmek ister. Bertil'in doğru tahmini yapmak için iki şansı vardır ve başarılı olursa hem o hem de Anna oyunu kazanır. Aksi takdirde, oyunu Emma kazanır.

Anna ve Bertil'in oyunu kazanmalarının garanti olduğu bir strateji geliştirebilir misin? Stratejiniz K değerine göre puanlanacaktır (ne kadar küçükse o kadar iyi).

Kodlama

Bu, çoklu çalıştırma gerektiren bir problemdir, yani programınız birden fazla kez çalıştırılacaktır. İlk kez çalıştırıldığında Anna'nın stratejisini uygulayacak. Dana sonra ise Bertil'in stratejisini uygulayacaktır.

Girdinin ilk satırı, P ve N olmak üzere iki tamsayı içerecektir. Burada P ya 1 ya da 2 olacaktır (birinci veya ikinci aşama), ve N ev sayısını belirtir. **Örnek girdiden farklı olarak (puanlama için kullanılmaz),** N **her zaman** 100000 **olacaktır**.

Aşağıdaki girdi aşamalara bağlıdır:

Aşama 1

Programınız K sayısını tek bir satıra $(1 \leq K \leq 1\,000\,000)$ yazarak başlamalıdır. Ardından, $N-1$ kez, i indisini $(0 \leq i < N)$ içeren bir satırı okumalı ve A_i tamsayılı $(1 \leq A_i \leq K)$ satırı çıktı olarak yazmalıdır. Burada A_i, Anna'nın i numaralı evin kapısına yazdığı sayıdır. Emma'nın evini gösteren gizli indis dışındaki her i indisi, grader tarafından kararlaştırılan bir sırayla olmak üzere tam olarak bir kez görünecektir.

Aşama 2

Programınız N tane tam sayı $A_0, A_1, \ldots, A_{N-1}$ içeren bir satırı okumalıdır. Burada, A_i, i numaralı evin kapısına yazılan sayıdır.

Ardından, tahmin edilen indisler olan s_1 ve s_2 $(0 \leq s_i < N)$ olmak üzere iki tamsayı içeren bir satır yazdırmalıdır. s_1 ve s_2'ın eşit olmasına izin verilir.

Kodlama Detayları

Programınız Aşama 2'de çalışırken programın yeniden başlatıldığını unutmayın. Yani, bu çalıştırmalar arasında bazı bilgileri değişkenler içinde kaydedemeyeceğiniz anlamına gelir.

Yazdırdığınız her satırdan sonra, standart çıktıyı temizlediğinizden emin olun, aksi takdirde programınız Zaman Sınırı Aşıldı (Time Limit Exceeded) sorunu ile karşılaşabilir. Python'da `print()` otomatik olarak temizler. C++'da, yeni satır yazdırmaya ek olarak `cout << endl;` da temizler; printf kullanıyorsanız `fflush(stdout)` kullanın.

Bu problem için grader uyarlanabilir olabilir, yani sezgisel (heuristic) çözümlerin geçmesini önlemek için programınızın çıktısına bağlı olarak davranışını değiştirebilir. Aşama 1'i çalıştırabilir, çıktınıza bakabilir ve ardından ilk çalıştırmadan çıkardığı bilgileri kullanarak aşama 1'i yeniden çalıştırabilir.

Programınız deterministik *olmalıdır*, yani aynı girdi üzerinde iki kez çalıştırılırsa aynı şekilde davranır. Programınızda rastgelelik olmasını istiyorsanız, sabit random seed kullandığınızdan emin olun. Bu programınıza `srand` (C++'da) veya `` `random.seed `` (Python'da) yazılarak yapılabilir. Veya, C++ 11'in rasgele sayı üreteçleri kullanılıyorsa, rasgele sayı üretecini oluştururken çekirdeği (seed) belirterek yapılabilir. Özellikle, C++'da `srand (time(NULL))` kullanamazsınız. Grader, programınızın deterministik olmadığını tespit ederse, Yanlış Cevap (Wrong Answer) kararı verilir.

Programınızın ayrı ayrı (3'e kadar) çalışmasının *toplamı* süre sınırını aşarsa, gönderiminiz Süre Sınırı Aşıldı (Time Limit Exceeded) olarak değerlendirilecektir.

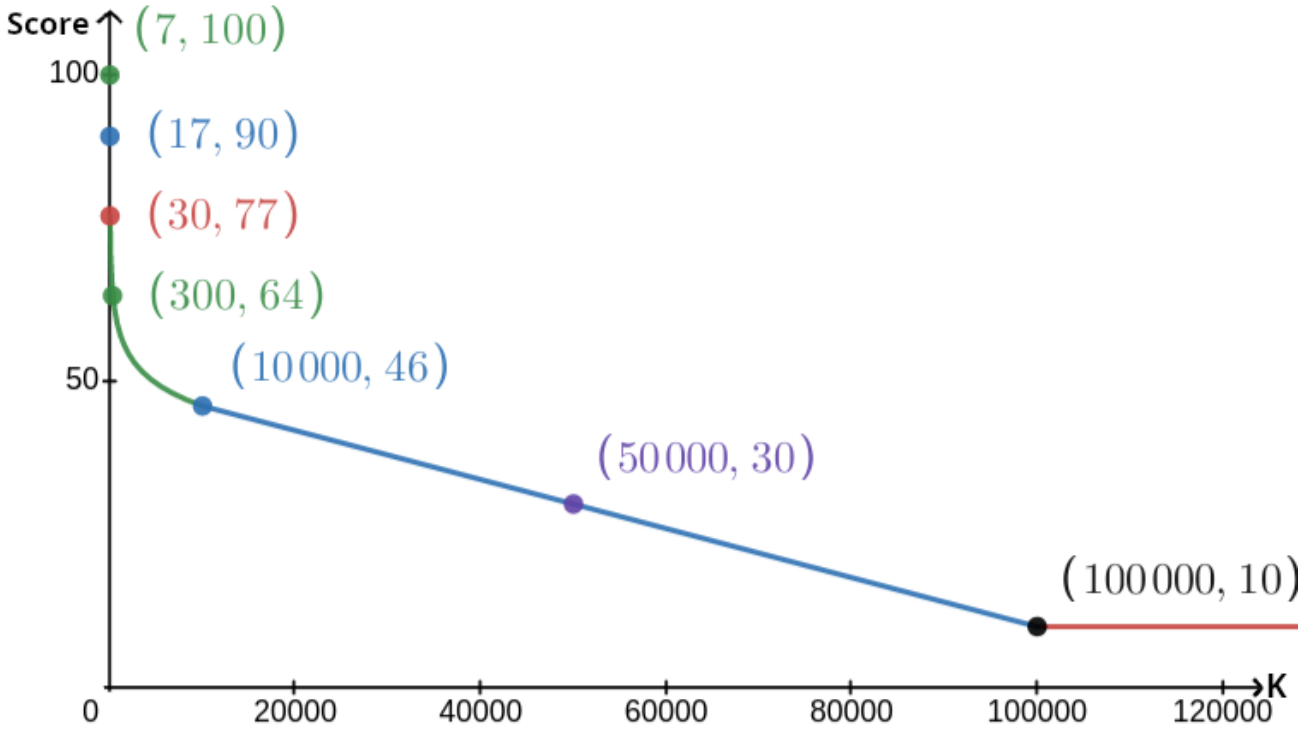
Puanlama

Çözümünüz bir dizi test senaryosu üzerinde test edilecektir. Çözümünüz bu test durumlarının *herhangi birinden* başarısız olursa (örneğin, yanlış cevaplar vermesi (Yanlış Cevap), göçmesi (Çalışma Zamanı Hatası), süre sınırını aşması (Zaman Sınırı Aşıldı), vb.), 0 puan alacaksınız ve buna uygun karar verilecektir.

Programınız *tüm* test senaryolarında Emma'nın evini gösteren gizli indisi başarıyla bulursa, Kabul edildi (Accepted) kararını alırsınız ve programınız şu şekilde hesaplanan bir puan alır. Herhangi bir test senaryosu için kullanılan maksimum K_{max} değeri K olsun. K_{max}'e bağlı olarak:

	Puan
$K_{max} > 99\,998$	10 puan
$10\,000 < K_{max} \leq 99\,998$	$10 + \lfloor 40(1 - K_{max}/10^5) \rfloor$ puan
$30 < K_{max} \leq 10\,000$	$46 + \lfloor 31(4 - \log_{10}(K_{max}))/(4 - \log_{10}(30)) \rfloor$ puan
$7 < K_{max} \leq 30$	$107 - K_{max}$ puan
$K_{max} \leq 7$	100 puan

Puanlama fonksiyonu aşağıdaki şekilde gösterilmektedir.



Örnek test senaryosu, puanlama için yoksayılır ve çözümünüzün bunun için çalışması gerekmez.

Test Aracı

Çözümünüzün test edilmesini kolaylaştırmak için indirebileceğiniz basit bir araç sunuyoruz. Kattis problem sayfasının altındaki "eklere" bakın. Aracın kullanımı isteğe bağlıdır ve onu değiştirmenize izin verilir. Kattis'teki resmi grader programının test aracından farklı olduğunu unutmayın.

Örnek kullanım ($N = 4$, $s = 2$, burada s ziyaret edilen son evin üzerine yazılan sayıdır):

Python için `solution.py` yazın (normalde `` `pypy3 solution.py `` olarak çalışır):

```
python3 testing_tool.py pypy3 solution.py <<<"4 2"
```

C++ için, ilk olarak derleyin (örneğin, `g++ -g -O2 -std=gnu++17 -static solution.cpp -o solution.out`) ve sonra çalıştırın:

```
python3 testing_tool.py ./solution.out <<<"4 2"
```

Test aracı evleri rasgele sırada ziyaret edecektir. Belirli bir sıra kullanmak için test aracını "MODIFY HERE" yazan yerde değiştirin.

Örnek Etkileşim

Örnek test senaryosu puanlama için yoksayılır ve çözümünüzün bunun için çalışması gerekmez.

$N = 4$ olduğunu ve Emma'nın 1 nolu evde yaşadığını varsayalım. Evlerin üzerine yazılan sayıların listesi A olsun. Başlangıçta, $A = [0, 0, 0, 0]$, burada 0, buna denk gelen evin üzerine herhangi bir sayı yazılmaması anlamına gelir.

Kodunuzun ilk çalışmasında:

$N = 4$ verilir. Çözümünüz $K = 3$ ile yanıt verir.

A_2 istenir. Çözümünüz 3 ile yanıt verir. A şimdi $[0, 0, 3, 0]$ olur.

A_0 istenir. Çözümünüz 1 ile yanıt verir. A şimdi $[1, 0, 3, 0]$ oldu.

A_3 istenir. Çözümünüz 2 ile yanıt verir. A şimdi $[1, 0, 3, 2]$ oldu.

Son olarak, grader $A_1 = 2$ değerini atar, böylece sonunda $A = [1, 2, 3, 2]$ olur. Bu, ilk aşamanın sonunu işaret eder.

Kodunuzun Aşama 2'sinde çözümünüz `1 2 3 2` listesi ile geçer.

`1 3` şeklinde yanıt verir.

Tahminlerden biri (1) evinin doğru indisi olduğu için Anna ve Bertil oyunu kazanır.

grader çıktısı	sizin çıktınız
1 4	
	3
2	
	3
0	
	1
3	
	2

grader çıktısı	sizin çıktınız
2 4	
1 2 3 2	
	1 3