Сүүлчийн оройн зоог

Леонардо өөрийнхөө хамгийн алдартай ханын усан будгийн зурган дээрээ ажиллаж байхдаа маш идэвхитэй байсан. Түүний өдөр тутмын даалгаврууд нэг нь ажлын өдрийн үлдсэн хугацаанд ямар өнгүүдийг ашиглах вэ? гэдэг байсан. Тэрэнд маш олон өнгүүд хэрэгтэй. (Ханын зураг зурахдаа зөөврийн шатан дээрээ будгаа тавьж зурдаг байсан. Цаашид зөөврийн шат гэхийг шат гэе.) Гэхдээ тэр шатан дээрээ зөвхөн хязгаартай тооны будаг л нэг дор байлгах боломжтой. Туслах нь агуулахаас шат руу хэрэгцээт нэг л будгийг зөөж аваачаад оронд нь өөр нэгэн будгийг агуулахад аваачдаг ажлыг хийх үүрэгтэй.

Энэ бодлогоны хувьд та Туслахад зориулсан хоёр тустай программ бичих ёстой. Эхний програм Леонардогийн зааврыг аваад (Өдрийн турш Леонардод хэрэг болох өнгөнүүдийн дараалал) мөн 'зөвөлгөө' гэдэг нэртэй *бага хэмжээний* төлөвлөгөө Туслахад зохиож өгөх ёстой. Өдрийн турш Туслах Леонардогийн хүсэлтүүдийг ашиглах боломжгүй бөгөөд эхний програмынхаа бий болгосон зөвөлгөөг ашиглана. Хоёр дахь програм зөвөлгөөг хүлээн авч мөн Леонардогийн хүсэлтийг тухай бүрд нь хүлээн авч ажиллана. Энэ програм нь зөвлөгөөг ямар утгатайг ойлгох ёстой бөгөөд хамгийн оновчтой сонголт хийхдээ ашиглах ёстой. Бүх зүйл доор илүү дэлгэрэнгүй тайлбарлагдсан байгаа.

Өнгийн будгийг агуулах болон шат хоёрын хооронд зөөх

Бид хялбаршуулсан хувилбарыг авч үзье. Энд N ширхэг будаг байгаа бөгөөд эдгээр нь 0-оос N-1-ийн хооронд дугаарлагдсан байгаа мөн өдөр бүр Леонардо яг N удаа будаг авахаар хүсэлт гаргадаг. C-г Леонардогийн гаргасан N будагны жагсаалт гэж үзье. Тиймээс бид C-г N тооны дараалал гэж үзнэ. Эдгээр тоонууд нь 0-оос N-1-ийн хооронд байна (0, N-1 нь орно). Зарим өнгөнүүд нь бүр дараалалд нэг ч удаа ороогүй байж болох бөгөөд мөн зарим нь олон удаа орсон байж болно гэдгийг анхаар.

Шат нь үргэлж N төрлийн будгаас аль нэг K ширхэг будгыг нь агуулж дүүрэн байх ёстой ($K < N$). Анхандаа шатан дээр 0-оос K-1 хүртэлх өнгөнүүд тавигдсан байна. (0 болон K-1 нь орно)

Туслах Леонардогийн хүсэлтийг нэг нэгээр нь биелүүлдэг. Хэрвээ хүссэн өнгийн будаг нь шатан дээр бэлэн байвал туслах амарч болно. Үгүй бол тэр Ленордагийн хүссэн өнгийн будгийг агуулахаас авчирч шатан дээр байрлуулах үүрэгтэй. Мэдээж шатан дээр илүү зай байхгүй тул туслах өмнө нь байсан будагнуудаас нэгийг нь сонгож аван буцаан агуулах руу байрлуулах хэрэгтэй болно.

Леонардогийн оновчтой стратеги

Туслах аль болох хамгийн олон удаа амрахыг хүсч байгаа. Түүний хэдэн удаа амарч чадах

нь хүсэлт биелүүлэх үедээ ямар сонголт хийхээс шууд хамаарна. Нарийвчлан хэлбэл шатнаас өнгө авч агуулах руу хураах бүртээ шатан дээрхи өнгөнүүдээс аль нэгийг нь сонгож болох бөгөөд сонголт бүр ирээдүйд ялгаатай үр дүнд хүргэнэ. Леонардо туслахдаа С-г/Леонардогийн тухайн өдөр гаргах хүсэлтүүд/ мэдсэнээр хэрхэн зорилгодоо хүрч болохыг зааж өгчээ. Агуулах руу хураах оновчтой сонгохын тулд шатан дээр одоо байгаа өнгөнүүд болон С-д үлдэж байгаа хүсэлтүүдийг харьцуулснаар хийж болно. Шатан дээрээс агуулах руу хураах өнгийг сонгохдоо хамгийн оновчтой сонголт хийхийн тулд дараахи стратегийг мөрдөх хэрэгтэй:

- Хэрвээ шатан дээр ирээдүйд дахин хүсэгдэхгүй өнгө байвал түүнийг агуулах руу хураана.
- Үгүй бол шатан дээрхи өнгийн будагнуудаас аль болох сүүлд хүсэгдэх өнгийг агуулах руу хураана.

Леонардогийн энэ стратегийг баримталбал туслах хамгийн их амарч чадна гэдгийг батлаж болох юм.

Жишээ 1

N = 4, гэж үзье. Бидэнд 0-оос 3 хүртэл дугаарлагдсан 4 өнгийн будаг болон 4 хүсэлт байгаа. Хүсэлт C = (2, 0, 3, 0) дарааллаар ирнэ гэж үзье. Мөн K = 2 гэж үзье. Энэ ньд Леонардогийн шат нь аль ч үед 2 өнгийн будаг багтаах зайтай гэсэн үг. Дээр дурьдсанчлан шатан дээр анх 0 болон 1 гэсэн өнгүүд байгаа. Бид шатан дээрх будгийг дараах байдлаар бичье: [0, 1]. Туслах эзнийхээ хүсэлтийг дараах байдлаар биелүүлж болно.

- Эхний хүссэн өнгийн будаг 2 нь шатан дээр байхгүй байна. Туслах үүнийг шатан дээр аваачиж тавиад 1-р өнгийн будгийг агуулах руу хураахаар шийдсэн. Одоо шатан дээр [0, 2] байна.

- Дараагийн хүссэн өнгийн будаг 0 нь аль хэдийн шатан дээр байгаа тул туслах амарч болно.

- 3-р хүссэн өнгийн будаг 3 нь шатан дээр байхгүй байна. Туслах үүнийг шатан дээр тавиад оронд нь 0-р өнгийн будгийг агуулах руу авч явахаар шийдсэн [3, 2].

- Эцэст нь сүүлийн хүссэн өнгийн будаг 0 нь шатан дээр байхгүй байгаа тул туслах 2-р өнгийн будгийг агуулахад аваачиж оронд нь 0-г тавихаар шийдсэн. Одоо шат [3, 0] боллоо.

Дээд талын жишээн дээр туслах Леонардогийн оновчит стратегийг ашиглаагүй гэдгийг анхаар. Оновчит стратеги нь 3-р алхам дээр 2-р өнгийн будгийг хураавал туслах сүүлийн алхам дээр мөн амрах боломжтой болно.

Туслахын санах ой хязгаарлагдмал үед баримтлах Туслахын стратеги

Туслах нь Леонардогаас С дарааллаа өглөө нь цаасан дээр бичээд үлдээхийг хүссэн тэгэснээр тэр оновчит стратегийг дагах боломжтой болно. Гэвч Леонардо нь өөрийн ажиллах техникээ нууц хэвээр нь байлгахыг хүсч байгаа тул туслахад цаас орхихоос татгалзсан. Тэр зөвхөн С дарааллыг нэг удаа уншаад дараа нь санах гэж оролдохыг нь зөвшөөрсөн.

Харамсалтай нь туслахын ой санамж маш муу. Тэр зөвхөн M хүртэлх бит санах чадвартай. Үүний улмаас тэр C хүсэлтийг бүхлээр нь санахгүй байж болно. Тиймээс тэр өөрийн тогтоож чадах M битний дарааллыг тооцоолох ухаалаг арга олох хэрэгтэй болсон. Тэр энэ дарааллаа *зөвлөх дараалал* гэж нэрлэн A гэж тэмдэглэсэн.

Жишээ 2

Өглөөд туслах C хүсэлт бүхий Ленардогийн цаасыг авч уншаад бүх шаардлагатай сонголтыг хийх боломжтой. Түүний хийж чадах нэг зүйл бол хүсэлт бүрийн дараахь шатны төлөв байдлыг шинжлэх юм. Жишээ нь жишээ 1-д дэд-оновчтой-шийдэл стратегийг ашиглахад шатны төлөв дараахи байдалтай байна. [0, 2], [0, 2], [3, 2], [3, 0]. Тэр шатны анхны төлөв болох [0, 1] мэднэ гэдгийг сана.

Одоо бидэнд M=16 байгаа гэж үзье. Энэ нь туслах 16 бит мэдээлэл санах боломжтой гэсэн үг. N=4 учраас бид өнгө бүрийг 2 битийн тусламжтайгаар хадгалж чадна. Тиймээс нийт 16 бит байхад дараахи дарааллыг хадгалахад хангалттай хангалттай. Тэгхээр Туслах дараахи 16 битийн урттай "зөвлөх дараалалыг үүсгэж болно" A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`).

Дараа нь туслах энэ зөвлөгөөний дарааллын кодыг тайлж уншин оновчтой сонголт хийхдээ ашиглах боломжтой.

Мэдээж M = 16 байхад Туслах оронд нь боломжит 16 bit-ээс зөвхөн 8-г ашиглан C дарааллаа бүтнээр санах сонголтыг хийж болно. Энэ жишээнд бид хамгийн сайн шийдийг хэлэхийн оронд зөвхөн тэрэнд бусад сонголтууд байж болох юмаа гэдгийг харуулахыг зорьсон.

Бодлогын даалгавар

Таны даалгавар бол 2 тусдаа програмыг ижил програмын хэлээр бичих юм. Эдгээр програмууд нь ажиллах явцдаа нэг нэгэнтэйгээ харьцах боломжгүйгээр дэс дарааллан биелнэх ёстой. Өөрөөр хэлбэл нэг програм ажилласны дараа нөгөөх нь ажиллана. Ажиллах явцдаа нэг нэгэндээ утга дамжуулгүй.

Эхнийхийг нь туслах өглөө л нэг удаа хэрэглэнэ. Үүнд C дараалал буюу өдөрийн турш ирэх хүсэлт өгөгдөхөд A буюу зөвлөх дарааллыг тооцоолох ёстой.

Хоёр дахь програмыг туслах өдрийн турш хэрэглэнэ. Энэ програм нь зөвлөгөөний дараалал болох A дараалыг аваад Леонардогийг хүсэлт болох C-г буцаах ёстой. Хүсэлтийн дараалал C энэ програмд нэг хүсэлт бүрд нэг удаа л мэдэгдэх боломжтой бөгөөд хүсэлт бүр дараагийн хүсэлт ирэхээс өмнө боловсруулагдах ёстой гэдгийг анхаар.

Бүүр тодорхой хэлбэл эхний програмд `ComputeAdvice`(C, N, K, M) гэсэн ганц функц бичих ёстой. Үүнд C нь N урттай бүхэл тоон массив. K нь шатны багтаамж, M нь зөвлөгөө өгөхөд хэрэглэгдэх битийн тоо. Энэ програм нь M битээс бүтэх зөвлөгөөний дараалал A-г боловсруулах ёстой. Ингэхдээ A дарааллын бит бүрийн хувьд дэс дарааллаар нь дараах функцыг дуудаж ажиллуулах ёстой.

- `WriteAdvice`(B) — нь бит B-г одоогийн зөвлөгөөний дараалалд залгана. Энэ функцыг хамгийн ихдээ M удаа дуудаж болно.

2 дахь програмд `Assist`(A, N, K, R) гэсэн нэг функцыг бичих ёстой. Энэ функцын оролт нь зөвлөгөөний дараалал болох A. Бүхэл тоо N, K нь дээр тодорхойлсны дагуу. A-гийн бодит урт R нь ($R \leq M$) байна. Энэ нь дараах функцуудыг ашиглан оновчтой стратегийг Туслахад ажилллуулж өгөх ёстой.

- `GetRequest`() - Леонардагийн хүсэж буй дараагийн өнгийг буцаана. Түүнээс хойшхи хүсэлтүүдийн талаар ямар нэг мэдээлэл илрэхгүй.

- `PutBack`(T) - T өнгийг агуулахад буцааж хураана. Энэ функцыг T өнгийн будаг шатан дээр байгаа үед л дуудаж ажиллуулах боломжтой.

Ажиллуулах үед `Assist` нь `GetRequest` функцыг яг N удаа дуудах ёстой. Дуудах бүртээ Леонардогийн нэг хүсэлтийг дарааллынх нь дагуу авна. `GetRequest` функцыг дуудсаны дараа хэрвээ хүссэн өнгө нь шатан дээр байхгүй бол `PutBack`(T) функцийг сонгосон T-гийн хамтаар мөн адил дуудах "ёстой". Үгүй бол `PutBack` функцыг дуудах ёсгүй. Ингэхгүй байх нь алдаа гэж үзэгдэх бөгөөд програмын ажиллагааг зогсооно. Эхлэх үед шатан дээр 0-оос K - 1 хүртэлхи (0 болон K-1 орно) өнгөнүүд байна гэдгийг анхаар.

Хэрвээ таны бичсэн програм заасан хязгаарлалтуудыг зөрчихгүй бөгөөд `PutBack` функыг дуудсан тоо нь яг Леонардогийн оновчтой шийдлийн тоотой ижил байвал бодсон гэж үзье. Тухайн тест дээр ижил тооны дуудалт хийдэг хэд хэдэн хамгийн сайн стратег байхын бол алийг нь ч гүйцэтгэсэн болно. Хэрвээ Ленордагийнхаас өөр оновчтой шийдэл байвал заавал Ленордаг дуурайх хэрэггүй.

Жишээ 3

2-р жишээг үргэлжлүүлье, `ComputeAdvice`-д та A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`) гэж олчихлоо гэж үзье. Үүнийг системд мэдээллэхын тулд та дуудалтуудыг дарааллан хийх ёстой: `WriteAdvice`(`0`) , `WriteAdvice`(`0`) , `WriteAdvice`(`1`), `WriteAdvice`(`0`) , `WriteAdvice`(`0`) , `WriteAdvice`(`0`) , `WriteAdvice`(`1`), `WriteAdvice`(`0`) , `WriteAdvice`(`1`) , `WriteAdvice`(`1`) , `WriteAdvice`(`1`), `WriteAdvice`(`0`) , `WriteAdvice`(`1`) , `WriteAdvice`(`1`) , `WriteAdvice`(`0`), `WriteAdvice`(`0`).

Таны дараагийн функц `Assist` нь дээрх A дараалал, мөн N = 4, K = 2, R = 16 өгөгдсөнөөр гүйцэлдэнэ. Функц `Assist` нь яг N = 4 удаа `GetRequest` дуудах ёстой. Мөн зарим хүсэлтүүдийн дараа `Assist` функц `PutBack`(T)-г тохиромжтой T тоотой цуг дуудах ёстой.

Доорх хүснэгтэнд дуудалтуудын дарааллыг жишээ 1 дээрх дэд-оновчтой-шийдэл сонголтын дагуу үзүүлсэн байна. Дундуур зураас `PutBack`-г дуудахгүй гэдгийг илтгэнэ.

GetRequest()	Үйлдэл
2	PutBack(1)
0	-
3	PutBack(0)
0	PutBack(2)

Дэд даалгавар 1 [8 оноо]

- $N \leq 5\,000$.
- Та хамгийн ихдээ M = 65 000 bit ашиглах боломжтой.

Дэд даалгавар 2 [9 оноо]

- $N \leq 100\,000$.
- Та хамгийн ихдээ M = 2 000 000 bit ашиглах боломжтой.

Дэд даалгавар 3 [9 оноо]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- Та хамгийн ихдээ M =1 500 000 bit ашиглах боломжтой.

Дэд даалгавар 4 [35 оноо]

- $N \leq 5\,000$.
- Та хамгийн ихдээ M = 10 000 bit ашиглах боломжтой.

Дэд даалгавар 5 [up to 39 оноо]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- Та хамгийн ихдээ M = 1 800 000 bit ашиглах боломжтой.

Дэд даалгаврын оноо нь R-ийн харьцаж байгаа зөвлөгөөний уртаар хэмжигдэнэ. Бүр нарийн хэлбэл хэрвээ R_{max} нь хамгийн их бол таны оноо `ComputeAdvice`:

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

Хэрэгжүүлэлтийн деталь

2 файлаа нэг програмчлалын хэл дээр бичих ёстой.

Эхний файл нь `advisor.c`, `advisor.cpp` эсвэл `advisor.pas` байна. Энэ файл нь `ComputeAdvice` функцыг тодорхойлсон байх ёстой бөгөөд `WriteAdvice` функцыг дуудсан байж болно. Хоёр дахь файл нь `assistant.c`, `assistant.cpp` эсвэл `assistant.pas` байх ёстой бөгөөд энэ файл нь `Assist` функцыг тодорхойлсон байх ёстой бөгөөд `GetRequest` болон `PutBack` функцыг ашигласан байж болно.

Бүх функцуудын тодорхойлолт дараах маягтай байна.

C/C++ програм

```
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 программ

```
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;
```

Гэхдээ таны бичсэн програм ямар нэг байдлаар стандарт оролт гаралт болон бусад файлтай харьцах ёсгүй.

Прогроамын шийд бичих үедээ дараах зааврууудыг мөн адил дагах ёстой.

C/C++ програм

Бодолтынхоо эхэнд `advisor.h` болон `assistant.h` 2-ийг advisor болон assistant дотор тус тус include хийх ёстой. Үүнийг кодондоо дараах мөрийг нэмснээр хийнэ.

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

эсвэл

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

`advisor.h` болон `assistant.h` нь шууд бодлогын хавтасанд байрласан байна. Ямар хэл ашигласнаас хамаарч `compile_c.sh` эсвэл `compile_cpp.sh` ажиллуулах ёстой.

Pascal програм

```
advisorlib болон assistantlib ийг ашиглах ёстой. Ингэхийн тулд дараахи кодыг бичнэ.
```

```
uses advisorlib;
```

эсвэл

```
uses assistantlib;
```

`advisorlib.pas` болон `assistantlib.pas` нь шууд бодлогын хавтсанд байрласан байна. Ямар хэл ашигласнаас хамаарч `compile_pas.sh` ажиллуулах ёстой.

Жишээ grader

Жишээ grader оролтыг дараах байдлаар уншина.

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

Grader эхлээд `ComputeAdvice` ажиллуулна. Энэ нь тусдаа битүүдийг агуулсан `advice.txt` файлыг үүсгэнэ. Энэ нь зайгаар тусгаарлагдсан 2-оор төгссөн байна.

Тэгээд `Assist`-ийг ажиллуулж эхлэнэ. Тэгээд мөр бүр нь `"R [number]"` ,`"P [number]"` эсвэл `"P [number]"` хэлбэртэй байх гаралт үүсгэнэ. Эхний төрлийн мөрүүд нь `GetRequest()`-ний дуудалтыг түүний хариутай агуулна. 2 дахь төрлийн мөрүүд нь `PutBack()` дуудалт болон буцааж тавихаар сонгосон өнгөнүүд байна. Гаралт нь `"E"` мөрөөр төгснө.

Цагтаа амжиж байгаа эсэхийг шалгахын тулд тестийн интерфэйсийг ашигла.

Санах ой болон хугацааны хязгаар

- Хугацааны хязгаарлалт: 7 seconds.
- Санах ойн хязгаарлалт: 256 MiB.