

Task 1. Hint

Doc Brown arriti ta kthente DeLorean-in e tij në një makinë kohe. Ai tani i ka vënë sytë te një problem edhe më i madh: nënsekuenca më e gjatë e zakonshme. Kur i jepen dy sekuenca të numrave A dhe B me gjatësi N dhe M , ai dëshiron të gjejë sekuencën më të gjatë C (ose një nga më të gjatat), të tillë që të gjithë elementët e C të shfaqen si në A ashtu edhe në B në të njëjtin rend (por jo domosdoshmërisht në mënyrë të njëpasnjëshme) si në C .

Ai ka arritur të shkruajë një program disi të ngadaltë i cili do të vazhdojë për shumë ditë derisa të gjejë një zgjidhje. Megjithatë, ai ka nevojë për një përgjigje sa më shpejt të jetë e mundur. Plani i tij fillestar ishte të linte programin e tij të funksiononte dhe më pas brenda pak ditësh ta dërgonte rezultatin e tij prapa në kohë tek vetja e tij aktuale. Problemi është se udhëtimi në kohë kërkon sasi të jashtëzakonshme energjie, kështu që dërgimi i zgjidhjes së plotë prapa në kohë do të ishte jashtëzakonisht i shtrenjtë. Tani Doc ka një plan të ri, por ai ka nevojë për ndihmën tuaj për ta zbatuar atë. Ai dëshiron të dërgojë një sugjerim të shkurtër për zgjidhjen nga e ardhmja në të tashmen dhe më pas përdorni këtë sugjerim për të rindërtuar një zgjidhje optimale duke përdorur këshillë. Shënim nuk duhet domosdoshmërisht të jetë e njëjta zgjidhje optimale si ajo e së ardhmes.

Ju duhet të shkruani një program `hint.cpp` i cili zbaton dy funksione: `genHint` dhe `solve`, të cilat arrijnë planin e Doc.

Detajet e implementimit

Funksioni juaj `genHint` duhet të ketë prototipet e mëposhtëm:

```
std::vector<bool> genHint(const std::vector<int>& a, const std::vector<int>& b,  
    const std::vector<int>& sol);
```

Ai do të thirret vetëm një herë dhe do të marrë si argumente dy sekuencat e dhëna dhe një zgjidhje optimale. Ai duhet të kthejë një `hint`, i cili do të dërgohet në funksionin tjetër.

Zgjidhja e funksionit tuaj duhet të ketë prototipin e mëposhtëm:

```
std::vector<int> solve(const std::vector<int>& a, const std::vector<int>& b,  
    const std::vector<bool>& hint);
```

Ai do të thirret vetëm një herë dhe do të marrë si argumente dy sekuencat e dhëna dhe `hint` që funksioni juaj tjetër kthen. Duhet të kthejë një zgjidhje optimale.

Përveç këtyre dy funksioneve, programi juaj mund të ketë çdo lloj funksioni ndihmës, klasa, variabla, etj. Megjithatë, ai **nuk duhet** të përmbajë një funksion kryesor dhe duhet të përfshijë header file `hint.h`.

Përveç këtyre dy funksioneve, programi juaj mund të ketë çdo lloj funksioni, klasa, variabla, etj. Megjithatë, ai nuk duhet të përmbajë një funksion kryesor dhe duhet të përfshijë `hint.h`. **Vini re se në sistemin e klasifikimit të dy funksionet tuaja do të thirren në instanca të veçanta të programit tuaj, duke ekzekutuar në procese të veçanta.** Kjo do të thotë se nuk do të jeni në gjendje të ndani asnjë global state midis ekzekutimeve të dy funksioneve.

Local testing

Ju jepet file `Lgrader.cpp`, të cilin mund ta kompiloni së bashku me programin tuaj për ta testuar. Do të lexohet N dhe M , e ndjekur nga A dhe B . Pas kësaj do të lexojë gjatësinë optimale të zgjidhjes K dhe zgjidhjen optimale të C . Ajo do të nxjerrë gjatësinë e `hint` nga funksionin `hint`, si dhe zgjidhjen që funksioni i zgjidhur kthen. Shënim, ndryshe nga sistemi zyrtar i notimit, notuesi lokal nuk i kryen funksionet tuaja në procese të veçanta.

Kufijtë

$$1 \leq N, M \leq 10^5$$
$$0 \leq A_i, B_j < \min(N, M)$$

Subtasks

Subtask	Points	N, M
1	10	$\leq 10^4$
2	90	$\leq 10^5$

Zgjidhja juaj do të marrë pikë për një subtask vetëm nëse i kalon të gjitha testet në të.

Scoring

Rezultati që merrni për një nëndetyrë të caktuar varet nga gjatësia maksimale L e një shënimi që programi juaj ka krijuar për një test në atë nëndetyrë. Pjesa e pikëve që merrni për nëndetyrën do të jetë:

$$\min\left(\left(\frac{640}{L+1}\right)^{0.3}, 1\right)$$