

Zadatak 1. Savjet

Doc Brown je uspio pretvoriti DeLoreana u vremensku mašinu. Sada želi da riješi još veći problem: Najduži zajednički niz (longest common sequence). Za data dva niza brojeva A i B sa dužinama N i M , želi da nađe najduži (ili jedan od najdužih) niz C , takav da svi elementi C se pojavljuju i u A i B u istom redoslijedu (ali ne nužno uzastopno) kao u C . Uspio je da napiše nešto spor program kojem treba nekoliko dana da nađe rješenje. Međutim, njemu treba rješenje mnogo brže od toga. Njegov inicijalni plan je bio da ostavi da program radi a onda za nekoliko dana da pošalje rezultat nazad u prošlost. Problem sa vremenskim putovanjem je to što treba nevjerovatno mnogo energije, tako da slanje čitavog rješenja bi bilo previse skupo.

Sada Doc ima novi plan, ali treba vašu pomoć za implementaciju. Želi da pošalje kratak "savjet" (hint) o rješenju iz budućnosti u prošlost a onda ga iskoristiti kako bi se rekonstruisalo optimalno rješenje uz pomoć tog savjeta. Primijetite da to rješenje nije nužno isto kao optimalno rješenje iz budućnosti.

Napišite program `hint.cpp` koje implementira dvije funkcije: `genHint` i `solve`, koje će postići Docov plan.

Detalji implementacije

Vaša funkcija `genHint` treba imati sljedeći prototip:

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

Ova funkcija će biti pozvana samo jednom i primiće kao parameter dva niza i optimalno rješenje. Ono treba da vrati savjet (hint), koji će biti poslan vašoj drugoj funkciji.

Vaša funkcija `solve` treba imati sljedeći prototip:

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

Ova funkcija će biti pozvana samo jednom i primiće kao parameter dva niza i savjet koji je vaša prva funkcija vratila. Ona treba vratiti optimalno rješenje.

Pored ove dvije funkcije vaš program može imati bilo kakve dodatne funkcije, klase, varijable, itd. Međutim **ne smije** sadržati `main` funkciju i **ne smije** da uključuje zaglavlje (header) `hint.h`. **Također, ove dvije funkcije grading system neće pozivati u istom procesu.** To znači da nećete moći dijeliti nikakve globalne varijable (ni slična stanja) između pokretanja ove dvije funkcije.

Lokalno testiranje

Dat vam je fajl `Lgrader.cpp`, koji možete kompajlirati zajedno s vašim programom da ga istestirate. Pročitajte N i M , a zatim A i B . Nakon toga će pročitati optimalno rješenje dužine K i optimalno rješenje C . Izbaciće kao rezultat dužinu savjeta iz vaše `hint` funkcije, kao i rješenje koje je vaša `solve` funkcija vratila. Primijetite da, za razliku od pravog grading systema, lokalni grader ne pokreće vaše funkcije u dva različita procesa.

Ograničenja

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

Podzadaci

Podzadatak	Bodovi	N, M
1	10	$\leq 10^4$
2	90	$\leq 10^5$

Vaše rješenje će dobiti bodove u podzadatku samo ako svi testovi u njemu budu tačni.

Bodovanje

Vaši bodovi za dati podzadatak će zavisi od maksimalne dužine L koju će vaša hint funkcija generisati za test u tom podzadatku. Dio bodova koje ćete dobiti za podzadatak bit će:

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