

Πρόβλημα 1. Βοήθεια (Hint)

Ο Δρ. Καφές κατάφερε να μετατρέψει την DeLorean του σε χρονομηχανή. Τώρα έχει στρέψει την προσοχή του σε ένα ακόμη μεγαλύτερο πρόβλημα, την εύρεση της μέγιστης κοινής υπακολουθίας. Όταν του δίνονται δύο ακολουθίες A και B , με μήκη N και M , θέλει να βρει την μακρύτερη ακολουθία C (ή μια από τις μακρύτερες αν υπάρχουν πολλές με το μέγιστο μήκος), έτσι ώστε όλοι οι αριθμοί της ακολουθίας C να εμφανίζονται στις ακολουθίες A και B με την ίδια σειρά όπως στην C (αλλά όχι απαραίτητα συνεχόμενα). Ο Δρ. Καφές κατάφερε να γράψει ένα πρόγραμμα για αυτό το πρόβλημα, αλλά το πρόγραμμά του είναι αρκετά αργό και θα κάνει μέρες μέχρι να βρει μια λύση. Ωστόσο, χρειάζεται την απάντηση όσο πιο γρήγορα γίνεται. Το αρχικό του πλάνο ήταν να αφήσει το πρόγραμμα να τρέχει και, σε μερικές μέρες που θα τελειώσει, να στείλει την λύση πίσω στον χρόνο στον εαυτό του. Το θέμα όμως είναι ότι το ταξίδι στον χρόνο απαιτεί τεράστια ποσά ενέργειας, οπότε το να στείλει πίσω ολόκληρη την λύση είναι πολύ ακριβό.

Ο Δρ. Καφές έχει ένα νέο σχέδιο, χρειάζεται όμως την βοήθειά σας για να το υλοποιήσει. Θέλει να στείλει μια μικρή βοήθεια για την λύση από το μέλλον στο παρόν, ώστε να τη χρησιμοποιήσει για να επανακατασκευάσει μια βέλτιστη λύση. Προσέξτε ότι δεν είναι απαραίτητο να κατασκευάσει την ίδια βέλτιστη λύση με εκείνη από το μέλλον.

Πρέπει να γράψετε ένα πρόγραμμα `hint.cpp` το οποίο θα υλοποιεί δύο συναρτήσεις, τις `genHint` και `solve`, οι οποίες θα βοηθήσουν τον Δρ. Καφέ να φέρει εις πέρας το σχέδιό του.

Λεπτομέρειες Υλοποίησης

Η συνάρτησή `genHint` πρέπει να έχει την ακόλουθη επικεφαλίδα:

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

Η συνάρτηση αυτή θα κληθεί μόνο μία φορά και θα παίρνει ως παραμέτρους τις δύο δοσμένες ακολουθίες και τη βέλτιστη λύση. Θα πρέπει να επιστρέφει μια βοήθεια, η οποία θα δοθεί στην άλλη συνάρτηση.

Η συνάρτηση `solve` πρέπει να έχει την ακόλουθη επικεφαλίδα:

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

Η συνάρτηση αυτή θα κληθεί μόνο μία φορά και θα παίρνει ως παραμέτρους τις δύο δοσμένες ακολουθίες και τη βοήθεια που επέστρεψε η άλλη σας συνάρτηση. Θα πρέπει να επιστρέφει μια βέλτιστη λύση.

Εκτός από αυτές τις δύο συναρτήσεις, το πρόγραμμά σας επιτρέπεται να χρησιμοποιεί οποιουδήποτε είδους βοηθητικές συναρτήσεις, κλάσεις, μεταβλητές, κτλ. Ωστόσο, **δεν πρέπει** να περιέχει μια συνάρτηση `main` και **πρέπει** να περιέχει το αρχείο επικεφαλίδων `hint.h`. **Προσέξτε ότι στο σύστημα ελέγχου οι δύο σας συναρτήσεις θα κληθούν σε ξεχωριστά στιγμιότυπα του προγράμματός σας και θα τρέχουν σε ξεχωριστές διεργασίες.** Αυτό σημαίνει ότι δεν θα έχετε τη δυνατότητα να μοιραστείτε καμία κατάσταση γενικής εμβέλειας μεταξύ τις εκτελέσεις των δύο συναρτήσεων.

Τοπικός έλεγχος

Σας δίνεται το αρχείο `Lgrader.cpp`, το οποίο μπορείτε να μεταγλωττίσετε μαζί με το πρόγραμμά σας για να το δοκιμάσετε. Το αρχείο αυτό θα διαβάζει τα μεγέθη των ακολουθιών N και M , ακολουθούμενα από τις ίδιες τις ακολουθίες A και B . Στη συνέχεια θα διαβάζει το μέγεθος της βέλτιστης λύσης K και την ίδια τη βέλτιστη λύση C . Τέλος θα εκτυπώνει το μέγεθος της βοήθειας από τη συνάρτηση `hint`, καθώς επίσης και τη λύση από τη

συνάρτηση `solve`. Προσέξτε ότι, σε αντίθεση με το επίσημο σύστημα ελέγχου, το τοπικό σύστημα δεν τρέχει τις συναρτήσεις σας σε ξεχωριστές διεργασίες.

Περιορισμοί

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

Υποπροβλήματα

Υποπρόβλημα	Πόντοι	N, M
1	10	$\leq 10^4$
2	90	$\leq 10^5$

Η λύση σας θα πάρει τους πόντους ενός υποπροβλήματος μόνο αν περάσει όλους τους ελέγχους σε αυτό.

Βαθμολόγηση

Οι πόντοι που λαμβάνετε για ένα υποπρόβλημα εξαρτιόνται από το μέγιστο μέγεθος L της βοήθειας την οποία έχει επιστρέψει το πρόγραμμά σας για όλους τους ελέγχους αυτού του υποπροβλήματος. Συγκεκριμένα το ποσοστό των πόντων που λαμβάνετε για το υποπρόβλημα είναι:

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