Collage di Arcobaleni (`collage` [90 punti])

Questo problema è preso ed adattato dalla Finale Nazionale 2004 delle Oii (Olimpiadi Italiane di Informatica).

Nel pianeta Wobniar ogni mattina splende un sorprendente arcobaleno i cui colori possono presentarsi anche più volte all'interno dell'arco in una sequenza sempre nuova e sorprendente. Ogni mattino all'alba l'artista Ed Esor cattura lo splendore del nuovo arco in un collage di strisce colorate. Per risparmiare sui materiali e meglio consentirne il riciclo, Ed Esor punta sempre a minimizzare il numero di fogli di carta colorata da sovrapporre nella composizione del collage, senza mai rinunciare a riprodurre fedelmente la sequenza apparsa in cielo. Ad esempio, se l'arcobaleno fosse composto da 3 strisce di 2 colori diversi alternati, basteranno allora due soli fogli di carta: uno, disposta come base, dello stesso colore delle due strisce alle estremità dell'arcobaleno, l'altro posato sul centro del primo.

Goals: con quanta efficienza sai rispondere alle seguenti domande?

[**opt_val**]: quale è il numero minimo di strisce per riprodurre l'arcobaleno

[**opt_sol**]: come disporre i fogli di carte colorata?

[**count_opts**]: quante sono le diverse soluzioni ottime?

Input

Si legga l'input da `stdin`. La prima riga contiene T, il numero di testcase (istanze) da risolvere. Seguono T istanze del problema, ciascuna descritta nel seguente formato: ad una prima riga contenente il numero n di strisce dell'arcobaleno segue una seconda riga di n numeri naturali $c_1, ..., c_n$ separati da spazi, con c_i a codificare il colore dell'i-esima striscia dell'arcobaleno.

Output

Per ciascuna istanza, prima di leggere l'istanza successiva, scrivi su `stdout` il tuo output così strutturato:

[**goal 1 (opt_val)**]: la prima riga contiene t, il minimo numero di fogli di carta da impiegare.

[**goal 2 (opt_sol)**]: la seconda riga contiene una sequenza ℓ di n numeri tutti presi nell'intervallo $[1, t]$: per ogni posizione $i = 1, ..., n$ nella sequenza il valore ℓ_i specifica il layer del foglio di colore c_i che appare in superfice nella striscia i-esima. Assumiamo che ciascuno dei t fogli impiegati venga collocato su un diverso layer (1 per il foglio collocato più in basso e t per il foglio che non può venire oscurato da nessun altro). Assumiamo inoltre che ogni foglio si estenda solo dalla prima all'ultima posizione in cui appare.

[**goal 3 (num_sols)**]: il numero di soluzioni ottime potrebbe essere molto grande. Calcola il resto della sua divisione per $1,000,000,007$.

Assunzioni

1. $1 \leq n \leq 200$
2. $1 \leq c_i \leq 100$

Esempio di Input/Output

Input da `stdin`

```
3
3
1 2 1
7
1 1 2 3 1 2 1
7
1 2 3 1 2 1 3
```

Output su `stdout`

```
2
1 2 1
1
4
1 1 3 4 1 2 1
3
5
1 4 5 1 3 1 2
7
```

Spiegazione: Nel primo caso di esempio i due corsi si sovrappongono nel giorno 5. Non essendo possibile frequentarli entrambi, conviene scegliere il secondo corso, per un valore totale di 2 crediti. Nel secondo caso di esempio una scelta ottima è frequentare solo il corso da 30 crediti; una seconda scelta ottima sarebbe frequentare i corsi 1 e 4. Nel terzo caso di esempio la soluzione ottima è nuovamente unica.

Subtask

Il tempo limite per istanza (ossia per ciascun testcase) è sempre di 1 secondo.

I testcase sono raggruppati nei seguenti subtask.

1. **[0 pts← 3 istanze da 0 + 0 + 0 punti] `esempi_testo`:** i tre esempi del testo
2. **[30 pts←10 istanze da 1 + 1 + 1 punti] `small`:** $n \leq 10$
3. **[30 pts←10 istanze da 1 + 1 + 1 punti] `medium`:** $n \leq 100$
4. **[30 pts←10 istanze da 1 + 1 + 1 punti] `big`:** $n \leq 1000$

In generale, quando si richiede la valutazione di un subtask vengono valutati anche i subtask che li precedono, ma si evita di avventurarsi in subtask successivi fuori dalla portata del tuo programma che potrebbe andare in crash o comportare tempi lunghi per ottenere la valutazione completa della sottomissione. Ad esempio, chiamando[1, 2] :

```
rtal -s <URL> connect -a size=medium collage -- <MY_SOLUTION>
```

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `small`, `medium`.

Il valore di default per l'argomento `size` è `big` che include tutti i testcase.

Nel template di riga di comando quì sopra `<MY_SOLUTION>` è una qualsiasi scrittura che ove immessa anche da sola al prompt della CLI comporti l'avvio del solver da tè realizzato. Solo alcuni esempi:

- `./a.out` per un compilato da `C/c++`, eventualmente seguito dagli argomenti che prevede
- `./my_solution.py arg1 arg2 ...` se il tuo file `my_solution.py` col codice python ha i permessi di esecuzione e inizia con la riga di shebang
- `python my_solution.py` o `python3 my_solution.py` per far eseguire il tuo script da un interprete python

Se vuoi che una tua sottomissione venga conteggiata ai fini di un esame o homework devi allegare il sorgente della tua soluzione con `-fsource=<FILENAME>` (ad esempio `-fsource=my_solution.py` subito a valle del comando `connect`. Inoltre devi aver precedentemente affettuato il login tramite credenziali GIA (lancia prima `rtal -s <URL> login` e segui le istruzioni per impostare il sign-on).

[1] `<URL>` server esame: `wss://ta.di.univr.it/esame`

[2] `<URL>` server esercitazioni e simula-prove: `wss://ta.di.univr.it/algo2025`

Il comando `rtal` prevede diversi parametri, consigliamo di esplorarne e sperimentarne le potenzialità ed opzioni d'uso. Un tutorial all'uso di `rtal` è esposto alla pagina:

`https://github.com/romeorizzi/AlgoritmiUniVR/tree/main/strumenti`