Abbinare le guide ai visitatori alla mostra (`mostra` [109 punti])

Questo problema è preso dalla fase territoriale delle Oii (Olimpiadi Italiane di Informatica).

All'ingresso della mostra sono presenti due file indiane: nella prima sono assiepati n turisti, nell'altra m studenti dell'Accademia delle Arti. Per una convenzione, tali studenti non abbisognano di alcun biglietto per entrare, basta il tesserino. Lo studente può entrare da solo oppure abbinato ad un turista il cui grado di competenza deve però essere inferiore a quello dello studente (altrimenti il Museo si fà una cattiva reputazione) Quando facciamo entrare un turita da solo il prezzo della visita è di 1 euro, mentre per le visite guidate il Museo quadagna 2 euro.

Per entrambe le file gli accessi avvengono uno alla volta e si è tenuti a rispettare l'ordine. Pertanto, se i e j sono turisti con $i < j$ non è possibile assegnare ad essi due studenti s_i ed s_j con $j \geq i$.

Con quanta efficienza sai rispondere alle seguenti domande?

[**opt_val**]: quale è il massimo ricavo € possibile per la Mostra espresso in euro? quale è il massimo numero di visite guidate che è possibile organizzare?

[**opt_sol**]: sai esibire un accoppiamento ottimo?

[**num_opt_sols**]: quanti sono gli accoppiamenti ottimi?

[**num_opt_sets**]: quanti sono gli insiemi di studenti di massima cardinalità che possano tutti spendersi contemporaneamente come guide?

Input

Si legga l'input da `stdin`. La prima riga contiene T, il numero di testcase (istanze) da risolvere. Seguono T istanze del problema, dove ogni istanza è descritta in 5 righe, di cui solo le prime 3 vengono trasmesse immediatamente. La prima contiene i tre numeri n ed m nell'ordine e separati da uno spazio. La seconda riga contiene gli n gradi di preparazione dei turisti, di nuovo separati da spazio e secondo l'ordine in cui accederanno al museo. La terza riga contiene gli m gradi di preparazione degli studenti, separati da spazio e secondo l'ordine in cui accederanno al museo.

Dopo che avrai già risposto in merito a quale sia il massimo guadagno ε ottenibile per il Museo, invieremo le restanti due righe. La quarta riga contiene un numero $r \geq 0$ inferiore al numero di set di studenti tutti accoppiabili. La quinta ed ultima riga contiene m valori 0/1 separati da spazi, dove gli uni individuano un sottoinsieme ottimo S di studenti che possano essere tutti impiegati come guide.

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 l'intero €, il massimo guadagno cui può ambire il Museo.

[**goal 2 (opt_sol)**]: le due righe seguenti codificano una soluzione riportando, nell'ordine e separati da spazio, i soli indici di quelle persone che entrano in coppia. La prima riga riporta i visitatori accompagnati da una guida e le seconda gli studenti che li accompagnano (non è richiesto che si curi l'allineamento).

[**goal 3 (num_opt_sols)**]: la riga successiva riporta il numero di accoppiamenti ottimi tra gli studenti e i turisti (questo numero può esplodere per istanze grandi, per evitare inefficienze noi smettiamo di computarlo non appena un qualche numero supera il $1,000,000,007$).

[**goal 4 (num_opt_sets)**]: la riga successiva riporta il numero di insiemi di studenti di massima cardinalità tra quegli insiemi per cui esista un accoppiamento che coinvolga come guide tutti gli studenti

contenuti. (Questo numero può esplodere per istanze grandi, dove comunque questo goal non viene conteggiato).

Assunzioni

1. $n, m \geq 1$
2. se i e j sono turisti con $i < j$ non è possibile assegnare ad essi due studenti s_i ed s_j con $j \geq i$
3. lo studente j può fare da guida al turista i dolo se $G[j] > V[i]$, dove $V[i]$ indica il grado di competenza dell'i-esimo turista nella fila dei turisti e $G[j]$ indica il grado di competenza del j-esimo studente nella fila degli studenti.

Esempio di Input/Output

```
<start in>
4
2 5
2 2
1 1 7 2 9
5 3
1 2 3 4 5
3 3 3
<more>
```

```
3 1
8 6 7
20
2 2
5 1
2 6

 <end>
```

```
<start out>
4
1 1
0 0 1 0 1
1
1
7
1 1 0 0 0
0 1 1
3
3
<more>
```

```
4
1 0 0
1
3
1
3
1 0
0 1
3
2

 <end>
```

Spiegazione: Nel primo caso d'esempio l'immagine seguente rappresenta l'unica soluzione ottima (che forma due coppie e ricava 4€):

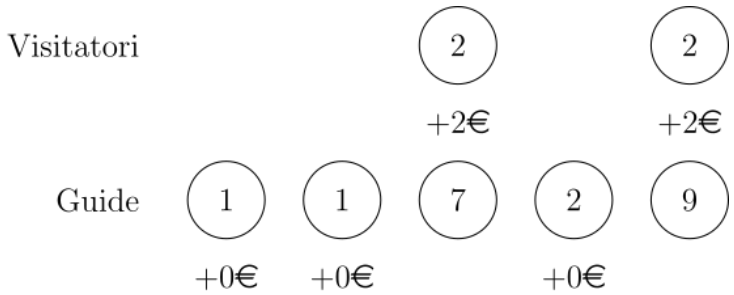

Figure 1: Configurazione iniziale dei computer e dei pulsanti.

Nel secondo caso d'esempio l'immagine seguente rappresenta l'unica soluzione ottima (che prevede due coppie e ricava 7€):

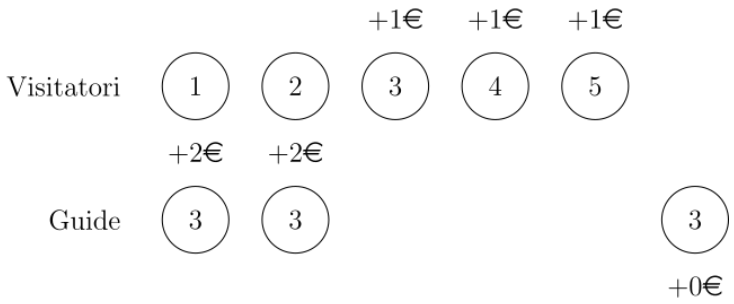

Figure 2: Configurazione iniziale dei computer e dei pulsanti.

Nel terzo caso d'esempio vi sono 3 accoppiamenti ottimi (prevedono una coppia e ricavano 4€):

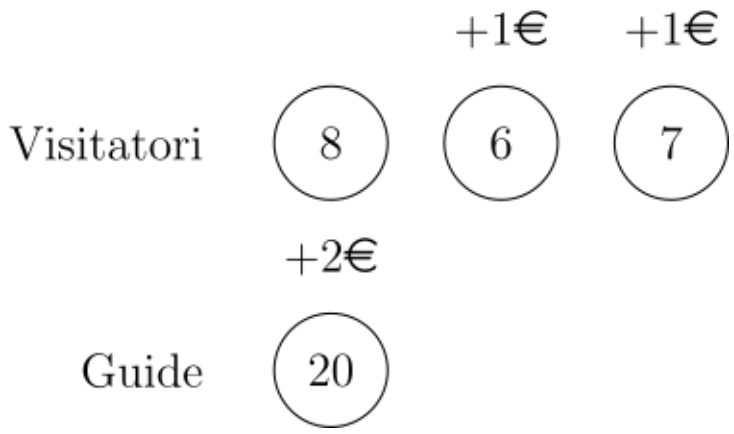

Figure 3: Configurazione iniziale dei computer e dei pulsanti.

Nel quarto caso d'esempio vi sono 3 accoppiamenti ottimi (prevedono una coppia e ricavano 3€)

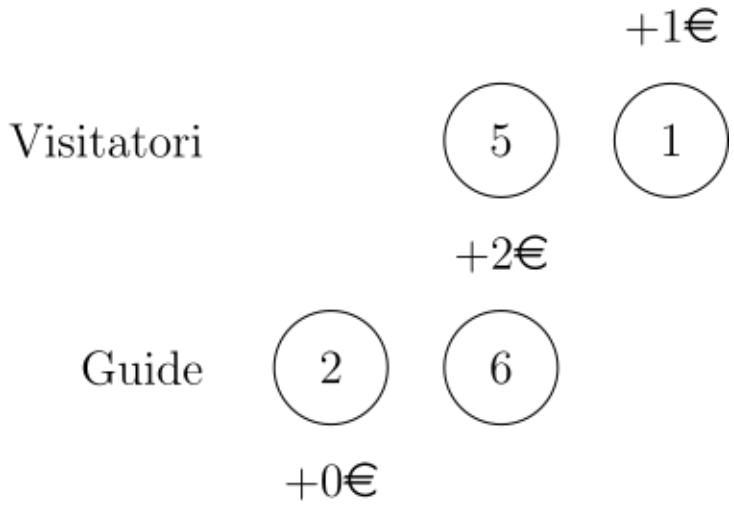

Figure 4: Configurazione iniziale dei computer e dei pulsanti.

Si noti come in questo ultimo esempio vi siano solo 2 sets ottimi di studenti mentre il numero di accoppiamenti ottimi è 3.

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← 4 istanze da 0 + 0 + 0 + 0 punti]** `esempi_testo`**:** i quattro esempi del testo
2. **[24 pts← 6 istanze da 1 + 1 + 1 + 1 punti]** `tiny`**:** $n \leq 10, m \leq 10$
3. **[20 pts← 5 istanze da 1 + 1 + 1 + 1 punti]** `small`**:** $n \leq 30, m \leq 30$
4. **[10 pts← 5 istanze da 1 + 1 + 0 + 0 punti]** `medium`**:** $n \leq 70, m \leq 70$
5. **[30 pts←15 istanze da 1 + 1 + 0 + 0 punti]** `big`**:** $n \leq 150, m \leq 150$
6. **[25 pts←25 istanze da 1 + 0 + 0 + 0 punti]** `large`**:** $n \leq 300, m \leq 300$

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] :

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

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

```
rtal -s <URL>  connect -x <token> -a size=small
       mostra  -- python my_solution.py
```

vengono valutati, nell'ordine, i subtask:

- `esempi_testo`, `tiny`, `small`.

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