Friend

Un social network è costruito a partire da n persone numerate da 0 a $n-1$. Alcune coppie di persone nel network saranno amiche, e se la persona x diventa amica della persona y, allora la persona y diventa amica della persona x.

Le persone sono aggiunte al network in n passi, che sono anche numerati da 0 a $n-1$, in cui la persona i è aggiunta al passo i. Al passo 0, la persona 0 è aggiunta come l'unica persona del network. In ciascuno dei successivi $n-1$ passi, la persona è aggiunta al network tramite un contatto *(host)*, che può essere una qualunque persona già nel network. Al passo i ($0 < i < n$), l'host per quel passo può aggiungere il nuovo arrivato i nel network seguendo uno dei seguenti tre protocolli:

- *IAmYourFriend* rende la persona i amica unicamente dell'host.
- *MyFriendsAreYourFriends* rende la persona i amica di *ogni* persona che al momento è amica dell'host. Notare che questo protocollo *non* rende la persona i amica dell'host.
- *WeAreYourFriends* rende la persona i amica dell'host, oltre che di *ogni* persona che al momento è amica dell'host.

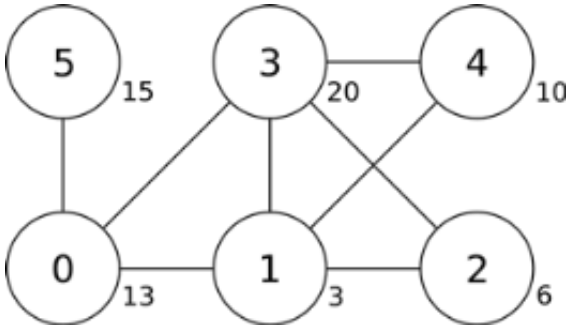
Dopo aver costruito il network vorremmo selezionare un campione *(sample)* per un sondaggio, e cioè, scegliere un gruppo di persone del network. Dato che solitamente amici tra loro hanno interessi simili, il campione non dovrebbe contenere nessuna coppia di persone che sono amici tra loro. Ogni persona ha un valore di confidenza *(confidence)* relativo al sondaggio, espresso come un intero positivo, e vorremmo trovare il campione dotato della massima confidenza totale.

Esempio

passo	host	protocollo	relazioni di amicizia aggiunte
1	0	IAmYourFriend	(1, 0)
2	0	MyFriendsAreYourFriends	(2, 1)
3	1	WeAreYourFriends	(3, 1), (3, 0), (3, 2)
4	2	MyFriendsAreYourFriends	(4, 1), (4, 3)
5	0	IAmYourFriend	(5, 0)

Inizialmente il network contiene unicamente la persona 0. L'host del passo 1 (la persona 0) invita la persona 1 tramite il protocollo IAmYourFriend, per cui i due diventano amici. L'host del passo 2 (di nuovo la persona 0) invita la persona 2 tramite MyFriendsAreYourFriends, il che rende la persona 1 (unico amico dell'host) l'unico amico della persona 2. L'host del passo 3 (la persona 1) aggiunge la persona 3 tramite WeAreYourFriends, il che rende la persona 3 amica sia della persona 1 (l'host) che di 0 e 2 (gli amici dell'host). I passi 4 e 5 sono ugualmente segnati nella tabella sopra.

Lo stato finale del network è mostrato nella figura seguente, in cui i numeri dentro ai cerchi indicano il numero identificativo delle persone, mentre i numeri a fianco ai cerchi mostrano la confidenza relativa al sondaggio. Il campione costituito dalle persone 3 e 5 ha confidenza totale pari a 20 + 15 = 35, che è il massimo possibile rispettando le richieste del problema.



Descrizione del problema

Data la descrizione di ogni passo e il valore di confidenza di ciascuna persona, trova un campione con la massima confidenza totale. Per fare questo devi implementare la sola funzione `findSample`.

- `findSample(n, confidence, host, protocol)`
  - `n`: numero di persone.
  - `confidence`: array di lunghezza n; `confidence[i]` rappresenta il valore di confidenza della persona i.
  - `host`: array di lunghezza n; `host[i]` rappresenta l'host del passo i.
  - `protocol`: array di lunghezza n; `protocol[i]` rappresenta il protocollo usato al passo i ($0 < i < n$): 0 per IAmYourFriend, 1 per MyFriendsAreYourFriends, e 2 per WeAreYourFriends.
  - Dato che non c'è nessun host al passo 0, `host[0]` e `protocol[0]` non sono definiti e quindi il vostro programma non dovrebbe farvi accesso.
  - La funzione dovrebbe restituire la massima possibile confidenza totale per un campione valido.

Subtask

Alcuni subtask usano unicamente un sottoinsieme dei protocolli possibili, come mostrato nella tabella seguente.

subtask	punti	n	confidenza	protocolli usati
1	11	$2 \leq n \leq 10$	$1 \leq \text{confidence} \leq 1\,000\,000$	Tutti i protocolli
2	8	$2 \leq n \leq 1\,000$	$1 \leq \text{confidence} \leq 1\,000\,000$	Solo MyFriendsAreYourFriends
3	8	$2 \leq n \leq 1\,000$	$1 \leq \text{confidence} \leq 1\,000\,000$	Solo WeAreYourFriends
4	19	$2 \leq n \leq 1\,000$	$1 \leq \text{confidence} \leq 1\,000\,000$	Solo IAmYourFriend
5	23	$2 \leq n \leq 1\,000$	Tutti i valori di confidence sono pari a 1	MyFriendsAreYourFriends e IAmYourFriend
6	31	$2 \leq n \leq 100\,000$	$1 \leq \text{confidence} \leq 10\,000$	Tutti i protocolli

Dettagli di implementazione

Devi sottoporre esattamente un file, chiamato `friend.c`, `friend.cpp` o `friend.pas`. Questo file deve implementare la funzione descritta sopra, usando l'intestazione seguente. In C/C++, devi anche includere il file header `friend.h`.

Linguaggio C/C++

```
int findSample(int n, int confidence[], int host[], int protocol[]);
```

Linguaggio Pascal

```
function findSample(n: longint, confidence: array of longint, host: array
of longint; protocol: array of longint): longint;
```

Grader di esempio

Il grader di esempio legge l'input secondo il formato seguente:

- riga 1: `n`
- riga 2: `confidence[0]`, ..., `confidence[n-1]`
- riga 3: `host[1]`, `protocol[1]`, `host[2]`, `protocol[2]`, ..., `host[n-1]`, `protocol[n-1]`

Il grader di esempio stamperà il valore restituito da `findSample`.