Freund

Wir erstellen ein soziales Netzwerk mit n Personen, welche mit 0, ... , $n-1$ durchnummeriert sind. Manche Paare von Personen im Netzwerk werden Freunde. Wann immer Person x ein Freund von Person y wird, wird auch Person y ein Freund von Person x.

Die Personen werden zum Netzwerk in n Schritten hinzugefügt, die ebenfalls mit 0 bis $n-1$ durchnummeriert sind. Person i wird in Schritt i hinzugefügt. In Schritt 0 wird Person 0 zum Netzwerk hinzugefügt (und ist dann die einzige Person im Netzwerk). In Schritt i ($0 < i < n$) wird Person i von einer schon hinzugefügten Person, der sogenannten *Kontaktperson*, hinzugefügt. Dies geschieht mittels eines der folgenden drei Protokolle:

- *IchBinDeinFreund* macht Person i zu einem Freund der Kontaktperson.
- *MeineFreundeSindDeineFreunde* macht Person i zu einem Freund *jeder* Person, die momentan mit der Kontaktperson befreundet ist. Beachte, dass dieses Protokoll Person i *nicht* zu einem Freund der Kontaktperson macht.
- *WirSindDeineFreunde* macht Person i zu einem Freund der Kontaktperson, sowie zu einem Freund *jeder* Person, die momentan Freund der Kontaktperson ist.

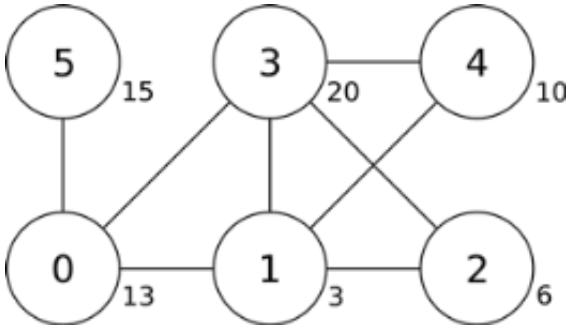
Nachdem das Netzwerk vollständig aufgebaut ist, möchten wir für eine Umfrage eine *Stichprobe* auswählen, d.h. eine Gruppe von Personen des Netzwerks. Da Freunde meist ähnliche Interessen besitzen, darf die Stichprobe keine zwei befreundeten Personen enthalten. Außerdem besitzt jede Person eine gewisse *Konfidenz*, eine positive ganze Zahl. Wir versuchen, eine Stichprobe mit maximaler Konfidenzsumme zu finden.

Beispiel

Schritt	Kontaktperson	Protokoll	Neue Freundschaften
1	0	IchBinDeinFreund	(1, 0)
2	0	MeineFreundeSindDeineFreunde	(2, 1)
3	1	WirSindDeineFreunde	(3, 1), (3, 0), (3, 2)
4	2	MeineFreundeSindDeineFreunde	(4, 1), (4, 3)
5	0	IchBinDeinFreund	(5, 0)

Zu Beginn enthält das Netzwerk nur Person 0. Die Kontaktperson in Schritt 1 (Person 0) fügt die neue Person 1 über das IchBinDeinFreund-Protokoll hinzu, sodass die beiden Freunde werden. Die Kontaktperson in Schritt 2 (wieder Person 0) fügt Person 2 über das MeineFreundeSindDeineFreunde-Protokoll hinzu, was Person 1 (den einzigen Freund der Kontaktperson) zum einzigen Freund von Person 2 macht. Die Kontaktperson in Schritt 3 (Person 1) fügt Person 3 durch das WirSindDeineFreunde-Protokoll hinzu, was Person 3 zu einem Freund von Person 1 (der Kontaktperson) und zu einem Freund der Personen 0 und 2 (der Freunde der

Kontaktperson) macht. Auch Schritte 4 und 5 sind in obiger Tabelle aufgeführt. Das resultierende Netzwerk ist in der folgenden Abbildung zu sehen, in der die umkreisten Zahlen die Nummern der Personen und die neben den Kreisen stehenden Zahlen die dazugehörigen Konfidenzen sind. Die aus Personen 3 und 5 bestehende Stichprobe besitzt eine Konfidenzsumme von 20 + 15 = 35, was auch die größtmögliche Konfidenzsumme ist.



Aufgabe

Gegeben die Beschreibung jedes Schrittes und die Konfidenz jeder Person, finde die maximale Konfidenzsumme einer Stichprobe. Du musst dazu nur die Funktion `findSample` implementieren.

- `findSample(n, confidence, host, protocol)`
  - `n`: die Anzahl Personen.
  - `confidence`: Array der Länge n; `confidence[i]` ist die Konfidenz von Person i.
  - `host`: Array der Länge n; `host[i]` ist die Nummer der Kontaktperson in Schritt i.
  - `protocol`: Array der Länge n; `protocol[i]` gibt das Protokoll für Schritt i $(0 < i < n)$ an: 0 für IchBinDeinFreund, 1 für MeineFreundeSindDeineFreunde, und 2 für WirSindDeineFreunde.
  - Da es in Schritt 0 keine Kontaktperson gibt, sind `host[0]` und `protocol[0]` nicht definiert und sollten von deinem Programm nicht verwendet werden.
  - Die Funktion soll die maximal mögliche Konfidenzsumme einer Stichprobe zurückgeben.

Subtasks

Einige Subtasks verwenden nur gewisse Protokolle, wie in folgender Tabelle angegeben.

Subtask	Punkte	n	Konfidenz	Benutzte Protokolle
1	11	$2 \leq n \leq 10$	$1 \leq \text{confidence} \leq 1\,000\,000$	Alle drei Protokolle
2	8	$2 \leq n \leq 1\,000$	$1 \leq \text{confidence} \leq 1\,000\,000$	Nur MeineFreundeSindDeineFreunde
3	8	$2 \leq n \leq 1\,000$	$1 \leq \text{confidence} \leq 1\,000\,000$	Nur WirSindDeineFreunde
4	19	$2 \leq n \leq 1\,000$	$1 \leq \text{confidence} \leq 1\,000\,000$	Nur IchBinDeinFreund
5	23	$2 \leq n \leq 1\,000$	Alle Konfidenzen sind gleich 1	Nur MeineFreundeSindDeineFreunde und IchBinDeinFreund
6	31	$2 \leq n \leq 100\,000$	$1 \leq \text{confidence} \leq 10\,000$	Alle drei Protokolle

Implementierungsdetails

Du musst genau eine Datei einsenden, mit Namen `friend.c`, `friend.cpp` oder `friend.pas`. Diese Datei muss die oben beschriebene Funktion mit der folgenden Signatur implementieren. Für C/C++-Implementierungen musst du außerdem die Headerdatei `friend.h` einbinden.

Programme in C/C++

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

Programme in Pascal

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

Sample-Grader

Der Sample-Grader liest die Eingabe in folgendem Format:

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

Der Sample-Grader gibt den Rückgabewert von `findSample` aus.