Friend

On construit un réseau social de n membres numérotés 0, ... , $n-1$. Certaines paires de membres du réseau seront amis. Si le membre x devient ami avec le membre y, alors le membre y est également ami avec le membre x.

Les membres sont ajoutés au réseau en n étapes qui sont également numérotées de 0 à $n-1$. Le membre i est ajouté à l'étape i. À l'étape 0, le membre 0 est ajouté comme unique membre du réseau. Dans chacune des $n-1$ étapes suivantes, un membre est ajouté au réseau par un *hôte* qui peut être n'importe quelle personne déjà membre du réseau. À l'étape i ($0 < i < n$), l'hôte de cette étape peut ajouter le membre i au réseau en suivant l'un des protocoles suivants :

- JeSuisTonAmi (*IAmYourFriend*) : fait de l'hôte l'unique ami du membre i.
- MesAmisSontTesAmis (*MyFriendsAreYourFriends*) : fait du membre i l'ami de *chaque* ami de l'hôte au moment de la requête. Ce protocole ne rend *pas* ce membre i ami de l'hôte.
- NousSommesTesAmis (*WeAreYourFriends*) : fait du membre i l'ami de l'hôte ainsi que de *chaque* ami de l'hôte au moment de la requête.

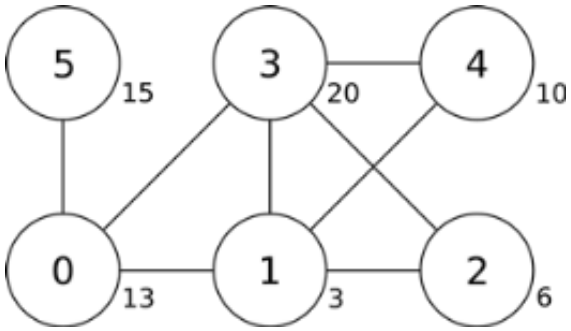
Après avoir construit le réseau, on aimerait pouvoir sélectionner un *échantillon* représentatif de personnes pour un sondage, c'est-à-dire un certain groupe de membres du réseau. Vu que des amis ont souvent les mêmes centres d'intérêt, l'échantillon ne doit pas inclure de paires de membres qui sont amis l'un de l'autre. Chaque membre a un niveau de *fiabilité* (`confidence`) pour le sondage, exprimé par un entier strictement positif. On aimerait trouver l'échantillon de membres qui maximise la fiabilité totale.

Exemple

étape	hôte	protocole	amitiés ajoutées
1	0	JeSuisTonAmi	(1, 0)
2	0	MesAmisSontTesAmis	(2, 1)
3	1	NousSommesTesAmis	(3, 1), (3, 0), (3, 2)
4	2	MesAmisSontTesAmis	(4, 1), (4, 3)
5	0	JeSuisTonAmi	(5, 0)

Initialement, le réseau ne contient que le membre 0. L'hôte de l'étape 1 (le membre 0) invite le nouveau membre 1 via le protocole JeSuisTonAmi, ils deviennent donc amis. L'hôte de l'étape 2 (le membre 0 de nouveau) invite le membre 2 via le protocole MesAmisSontTesAmis, qui fait du membre 1 (l'unique ami de l'hôte) l'ami (le seul) du membre 2. L'hôte de l'étape 3 (le membre 1) ajoute le membre 3 via le protocole NousSommesTesAmis, qui fait du membre 3 l'ami du membre 1 (l'hôte) et des membres 0 et 2 (amis de l'hôte). Les étapes 4 et 5 sont également données dans la table ci-dessus. Le réseau final est représenté dans le schéma suivant dans lequel les nombres à l'intérieur des cercles

indiquent le numéro du membre et les nombres à côté des cercles indiquent le niveau de fiabilité. L'échantillon composé des membres 3 et 5 a un niveau total de fiabilité égal à 20 + 15 = 35, qui est le niveau total de fiabilité maximal qu'il est possible d'atteindre.



Tâche

Étant donnés la description de chaque étape et le niveau de fiabilité de chaque membre, trouvez l'échantillon avec le niveau total maximal de fiabilité. Vous devez uniquement implémenter la fonction `findSample`.

- `findSample(n, confidence, host, protocol)`
  - `n` : le nombre de membres.
  - `confidence` : un tableau de taille n ; `confidence[i]` indique la fiabilité du membre i.
  - `host` : tableau de taille n ; `host[i]` indique l'hôte de l'étape i.
  - `protocol` : tableau de taille n ; `protocol[i]` indique le code du protocole utilisé pour l'étape i ($0 < i < n$) : 0 pour JeSuisTonAmi, 1 pour MesAmisSontTesAmis, et 2 pour NousSommesTesAmis.
  - Vu qu'il n'y a pas d'hôte à l'étape 0, `host[0]` et `protocol[0]` sont indéfinis et ne doivent pas être utilisés par votre programme.
  - La fonction doit retourner le niveau total maximal possible de fiabilité.

Sous-tâches

Certaines sous-tâches utilisent un sous-ensemble de protocoles, comme défini ci-dessous.

sous-tâche	points	n	fiabilité	protocoles utilisés
1	11	$2 \leq n \leq 10$	$1 \leq$ fiabilité $\leq 1.000.000$	Les 3 protocoles
2	8	$2 \leq n \leq 1.000$	$1 \leq$ fiabilité $\leq 1.000.000$	Uniquement MesAmisSontTesAmis
3	8	$2 \leq n \leq 1.000$	$1 \leq$ fiabilité $\leq 1.000.000$	Uniquement NousSommesTesAmis
4	19	$2 \leq n \leq 1.000$	$1 \leq$ fiabilité $\leq 1.000.000$	Uniquement JeSuisTonAmi
5	23	$2 \leq n \leq 1.000$	Toutes les fiabilités valent 1	MesAmisSontTesAmis et JeSuisTonAmi
6	31	$2 \leq n \leq 100.000$	$1 \leq$ fiabilité ≤ 10.000	Les 3 protocoles

Implémentation

Vous devez soumettre un seul fichier, appelé `friend.c`, `friend.cpp` ou `friend.pas`. Ce fichier doit implémenter la fonction décrite précédemment, en respectant les signatures suivantes. Vous devez inclure (`#include`) l'en-tête `friend.h` pour l'implémentation C/C++.

Programme C/C++

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

Programme Pascal

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

Évaluateur

L'évaluateur fourni lit l'entrée dans le format suivant :

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

L'évaluateur affichera la valeur retournée par `findSample`.