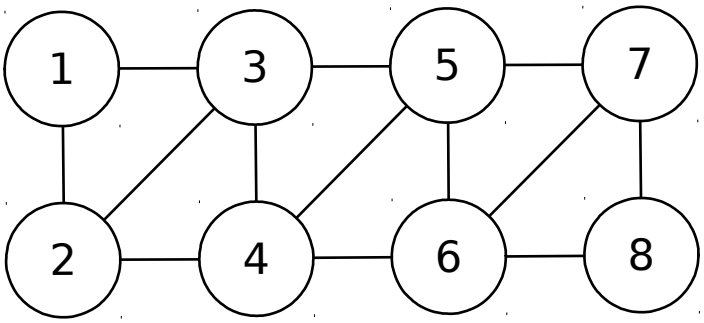
Վասին վերջապես որոշեց այցելել Ռուսե և զբոսնել այնտեղի զբոսայգում։ Իր գիդից նա իմացել է, որ զբոսայգին ուղղանկյան տեսք ունի և նրա երկար կողմի երկարությունը N միավոր է։ Երկու երկար կողմերի երկայնքով կան $N+1$ հետաքրքիր վայրեր (սրճարաններ, թենիսի դաշտեր, շատրվաններ և այլն), այսպիսով ընդամենը $2N+2$ հատաքրքիր վայր կա։ Նրանք համարակալված են և միացված են ծառուղիներով, ինչպես ցույց է տրված ստորև նկարում ($N=3$ դեպքում)։ Զբոսայգին ունի մեկ մուտք (1 համարի վայրը) և մեկ ելք ($2N+2$ համարի վայրը)։

Բոլոր ծառուղիներն այգում միակողմանի են և քայլելուց պետք է գնալ միայն յուրաքանչյուր ծառուղու ուղղությամբ։ Ծառուղիներն այնպես են կառուցված, որ **ցիկլեր չկան:** Նաև, մուտքը միակ վայրն է, **որտեղ բոլոր ծառուղիները դուրս եկող են**, և ելքը միակ վայրն է, **որտեղ բոլոր ծառուղիները մտնող են:** Մնացած բոլոր վայրերի



համար տեղի ունի հետևյալ կանոնը. **կան և՛ մտնող, և՛ դուրս եկող ծառուղիներ:**

Կասենք ծառուղին ունի դրական ուղղություն, եթե այն տանում է ավելի փոքր համարով վայրից դեպի ավելի մեծ համարով վայրը, հակառակ դեպքում կասենք, որ նրա ուղղությունը բացասական է։ Վասին տեղյակ է, որ մուտքից կա դուրս եկող երկու դրական ծառուղի և կա ելք տանող երկու դրական ծառուղի (քանի որ մուտքի համարը 1 է, իսկ ելքի համարը՝ $2N+2$), բայց նա ոչինչ չգիտի մյուս ծառուղիների ուղղությունների մասին, իսկ նրան դա պետք է իր զբոսանքը պլանավորելու համար։

Մաթեմատիկայի և ինֆորմատիկայի խնդիրների սիրահարների համար զբոսայգու մուտքի մոտ կա հատուկ ծրագրով համակարգիչ։ Այն պատասխանում է հատկյալ տիպի հարցերի՝ ծրագրին տրվում է ծառուղիների կամայական ցուցակ, ներկայացված իրենցով միացվող վայրերի համարներով, և ծրագիրը վերադարձնում է նրանց ուղղությունների վրա կիրառված XOR գործողության արդյունքը (որտեղ դրական ուղղությունը ներկայացվում է 1-ով, իսկ բացասականը՝ 0-ով)։ *Հիշեցնենք, որ երկու բիթերի XOR-ը 1 է, եթե նրանք տարբեր են, և 0 է, եթե նրանք նույնն են։ Երկուսից ավել օպերանդների դեպքում մենք սկզբում կիրառում ենք XOR-ը առաջին երկուսի վրա, հետո նրանց արդյունքի և երրորդ օպերանդի վրա և այդպես շարունակ։* **Ծրագիրը գրված է այնպես, որ մեկ ծառուղի տալու դեպքում պարզապես վերադարձնում է նրա ուղղությունը:**

Վասին ցանկանում է պարզել բոլոր ծառուղիների ուղղությունները առանց ծրագրին շատ հարցեր տալու։

Ժյուրին պատճենել է ծրագիրը զբոսայգու մուտքի մոտից և հիմա այն բեռնված է ստուգող համակարգում։ Օգնեք Վասիին գրելով ֆունկցիա, որը կկոմպիլացվի ժյուրիի ծրագրի հետ և կհաղորդակցվի նրա հետ վերը նկարագրված տիպի հարցեր տալով և ֆիքսելով իր գտած ուղղությունները։ Իր աշխատանքն ավարտելուց հետո ձեր ֆունկցիան պետք է ունենա զբոսայգու բոլոր ծառուղիների ճիշտ գտնված և ֆիքսված ուղղությունները։

Իրականացման մանրամասներ

Ձեր ֆունկցիան պետք է ունենա այսպիսի նախատիպ.

```
void run(int n);
```

Այն կանչվելու է միայն մեկ անգամ և որպես արգումենտ ստանալու է զբոսայգու երկար կողմի N երկարությունը, որը դրական ամբողջ թիվ է (հետաքրքիր վայրերի ընդհանուր քանակը $2N+2$) է։

Ժյուրիի ծրագրի հետ հաղորդակցվելու համար կարող եք օգտագործել հետևյալ երկու ֆունկցիաները.

```
bool get_xor(const std::vector<std::pair<int, int>>& edges);
void state_direction(const std::pair<int, int>& edge, bool direction);
```

`get_xor`-ի յուրաքանչյուր կանչի դեպքում կվերադարձվի վեկտորում պահվող ծառուղիների ուղղությունների XOR գործողության արդյունքը։ Յուրաքանչյուր ծառուղի ներկայացված է իրենով կապվող երկու հետաքրքիր վայրերի զույգի տեսքով (irrespective of order). Նկատենք, որ `get_xor` ֆունկցիայի բարդությունը (և՛ ժամանակի, և՛ հիշողության համար) գծային է կախված հարցում ծառուղիների քանակից։

Ձեր ֆունկցիան պետք է կանչի `state_direction` ֆունկցիան յուրաքանչյուր ծառուղու համար (կրկին ներկայացված նույն եղանակով) գտնված ուղղությունով, որը տրվում է `direction` պարամետրի միջոցով։ Եթե որևէ պահի ձեր ծրագիրը կանչի նշված երկու ֆունկցիաներից որևէ մեկը սխալ ծառուղով կամ `state_direction`-ը սխալ ուղղությունով, դուք այդ թեստի համար կստանաք զրո միավոր։ Դուք թեստի համար զրո միավոր կստանաք նաև այն դեպքում, եթե լինեն ծառուղիներ, որոնց համար ձեր ֆունկցիան չի կանչել `state_direction`-ը։

Ստուգող համակարգին ուղարկեք **park.cpp** ֆայլը։ Նրանում բացի նշված ֆունկցիայից կարող եք ունենալ այլ օժանդակ ֆունկցիաներ, փոփոխականներ, դասեր և այլն։ Սակայն այնտեղ չպետք է լինի `main` ֆունկցիա և այնտեղ պետք է միշտ ընդգրկված լինի `park.h` ֆայլը պրեպրոցեսորի `#include "park.h"` հրահանգով ֆայլի սկզբում։

Սահմանափակումներ

$1 \le N \le 100000$

Գնահատումը

Ձեր ծրագիրը կստանա 0-ից տարբեր միավոր, եթե յուրաքանչյուր թեստի համար այն հաջողությամբ կատարում է խնդիրը տրված ժամանակ սահմանափակումով և յուրաքանչյուր թեստի համար `get_xor` ֆունկցիային տրված հարցերի քանակը չի գերազանցում $3N$-ը։ Միավորի հաշվարկը կարվի հետևյալ կերպ (Q – ն մեկ թեստում `get_xor`-ի կանչերի քանակն է; $\lceil x \rceil$ –ը ամենափոքր ամբողջ թիվն է $\ge x$)։

15 միավոր. $Q \le 3N$ բոլոր թեստերի համար, բայց $Q > 2N$ նույնիսկ մեկ թեստի համար։

45 միավոր. $Q \le 2N$ բոլոր թեստերի համար, բայց $Q > \lceil \frac{3}{2} N \rceil$ նույնիսկ մեկ թեստի համար։

70 միավոր. $Q \le \lceil \frac{3}{2} N \rceil$ բոլոր թեստերի համար, բայց $Q > \lceil \frac{16}{11} N \rceil$ նույնիսկ մեկ թեստի համար։

100 միավոր. $Q \leq \lceil \frac{16}{11} N \rceil$ բոլոր թեստերի համար։

Լոկալ թեստավորում

Ձեզ տրված է `Lgrader.cpp` ֆայլը, որը դուք կարող եք կոմպիլացնել ձեր ծրագրի հետ և թեստավորել այն։ Աշխատացնելուց այն ձեզ կհարցնի N-ը և դրանից հետո բոլոր ծառուղիների ուղղությունները մեկը մյուսի ետևից (0 կամ 1)։ Այն կտպի տեղի ունեցող հաղորդակցությունները։ Դուք կարող եք ձևավորել այս ֆայլը ինչպես կամենաք։

Թեստավորումը համակարգում և adaptive թեստեր

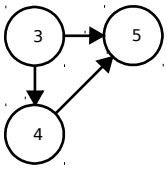
Որոշ (ձեզ համար անհայտ) թեստերում ժյուրիի ծրագիրը հարմարվելու է ձեր ծրագրի տված հարցերին, այսինքն ծառուղիների ուղղությունները կարող են փոխվել կախված ուղարկված հարցերից։ Եթե դուք թեստավորեք ձեր ծրագիրը ստուգող համակարգում adaptive օպցիան ձեզ հասանելի չի լինի։ Մուտքային ֆայլի ձևաչափը պետք է լինի այսպիսին. N բնական թիվ, ապա $4N+1$ զրոներ և մեկեր՝ ծառուղիների ուղղությունները մեկը մյուսի ետևից (1-2, 1-3, 2-3, 2-4, ..., k-(k+1), k-(k+2), ...)։

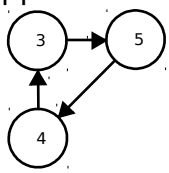
Հաղորդակցման օրինակ

№	Ձեր ֆունկցիայի գործողությունները	Ժյուրիի ծրագրի գործողությունները և պատասխանները
1.		`run(2)`
2.	`state_direction({1,2}, 1)`	
3.	`state_direction({1,3}, 1)`	
4.	`state_direction({4,6}, 1)`	
5.	`state_direction({5,6}, 1)`	
6.	`get_xor({{3,4},{3,5},{4,5}})`	`1`
7.	`get_xor({{3,5}})`	`1`
8.	`state_direction({3,4}, 1)`	
9.	`state_direction({3,5}, 1)`	
10.	`state_direction({4,5}, 1)`	
11.	`get_xor({{2,3},{2,4}})`	`0`
12.	`state_direction({2,3}, 1)`	
13.	`state_direction({2,4}, 1)`	
14.	The function terminates.	

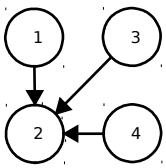
Հաղորդակցման օրինակի բացատրությունը

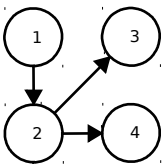
1. Ժյուրիի ծրագիրը կանչում է `run`-ը $N=2$-ով:
2-5. Տրվում են մուտքից դուրս եկող և ելք մտնող ծառուղիների ուղղությունները:
6-7. Երկու XOR-ների հարցում է արվում և երկուսի համար էլ ստացվում է 1: Եկեք քննարկենք ծառուղիների երկու հնարավորությունները:





8-10. Երկրորդ տարբերակը ցիկլ է պարունակում, բայց զբոսայգում ցիկլեր չկան, հետևաբար առաջին տարբերակի ուղղություններն են տրվում:
11. Կանչում հարցում է արվում երկու ուղղությունների XOR-ների վերաբերյալ և ստացվում է 0: Հետևաբար, նրանց ուղղությունները նույնն են.





12-13. Առաջին տարբերակում 2-ում միայն մտնող ծառուղիներ կան, բայց այդպիսի միայն մեկ վայր կա զբոսայգում (ելքը) իսկ սա ելքը չէ, այսպիսով ընտրվում է երկրորդ տարբերակը:
14. Ֆունկցիան ավարտվում է: 3 հարցում է օգտագործվել: