به نام خدا

آزمون سوم
اردیبهشت ۱۳۸۶

تونل‌های کپل
(Tunnels)

زمان مجاز برای هر مورد: ۱۰۰۰ میلی‌ثانیه[۱]
حافظه‌ی مصرفی مجاز: ۶۳ مگابایت[۲]

کُپل: دم‌باریییک! دم‌باریک جونم! بیا ... بیاااا ...
دُم‌باریک: هان. چـــیه؟
کپل: ببییین. اون n تا انبار گردو بود که پیدا کردیم.
دم‌باریک: خوب؟
کپل: من بینشون یه سری تونل کشیدم. الان می‌شه با این تونل‌ها از هر انبار به هر انبار دیگه‌ای رفت.
دم‌باریک: راست می‌گی کپل جونم؟ تو که خیلی اهل تونل کشیدن و این جور زحمت‌ها نبودی.
کپل: خوب، من تونستم فقط با $n-1$ تونل این کار رو انجام بدم.
دم‌باریک: کپل‌جونم، می‌شه بگی چه انبارهایی رو مستقیماً با تونل به هم وصل کردی؟
آقا معلم: کپل! چه قدر حرف می‌زنی؟!
کپل: هیسس. من نمی‌تونم خیلی حرف بزنم. ولی اگر به این شکل سؤال کنی، می‌تونم جوابت رو بدم: «تو چندتا انبار را بگو، من تعداد تونل‌هایی رو می‌نویسم که هر دو سرشون به انبارهایی ختم می‌شه که تو گفتی.»

می‌دانیم دو سرِ هر تونل به یک انبار ختم می‌شود و تونل‌ها با هم تقاطع ندارند مگر در سرهایشان. به دم‌باریک کمک کنید تا با تعداد کمی سؤال بفهمد $n-1$ تونلِ کپل چه انبارهایی را مستقیماً به هم وصل کرده‌اند. همچنین می‌دانیم کپل حوصله‌ی پاسخ دادن به بیش از ۲۰۰۰۰ سؤال را ندارد.

مسئله

برنامه‌ای بنویسید که

- تعداد انبارها (n) را از **کتاب‌خانه‌ی پیوندی** بخواند؛
- به کمک **کتاب‌خانه‌ی پیوندی** سؤالات مورد نظر را از کپل بپرسد؛
- به **کتاب‌خانه‌ی پیوندی** گزارش دهد چه انبارهایی با تونل مستقیماً به هم راه دارند.

[۱] زمان مجاز در واقع ۱۵۰۰ میلی‌ثانیه می‌باشد و کتاب‌خانه حداکثر ۵۰۰ میلی‌ثانیه از آن را استفاده می‌کند.
[۲] حافظه‌ی مصرفی مجاز در واقع ۶۴ مگابایت می‌باشد و کتاب‌خانه حداکثر ۱ مگابایت از آن را استفاده می‌کند.

محدودیت‌ها

$۱ \le n \le ۱۰۰۰$ و در ۲۵٪ تست‌ها $n \le ۲۵۰$.

کتاب‌خانه

سه تابع زیر در کتاب‌خانه یافت می‌شوند:

`int init()` این تابع n را باز می‌گرداند.

`int countTunnels(const bool *q)` این تابع یک سؤال از کپل می‌پرسد و پاسخ او را باز می‌گرداند. پارامتر ورودی این تابع، اشاره‌گری است به ابتدای یک آرایه به طول n از نوع bool. خانه‌ی i اُم این آرایه را true قرار دهید اگر و تنها اگر می‌خواهید انبار i ام را در این سؤال انتخاب کنید. توجه کنید که `q[0]` مربوط به انبار ۱، و `q[n-1]` مربوط به انبار n است.

`void report(const int *a, const int *b)` با فراخوانی این تابع می‌توانید پاسخ خود را به کتاب‌خانه تحویل دهید. این تابع باز نمی‌گردد و پس از فراخوانی‌اش برنامه‌ی شما پایان می‌یابد. ورودی این تابع، دو اشاره‌گر است به ابتدای آرایه‌هایی به طول $n-۱$ از نوع int. در خانه‌ی i اُم این دو آرایه، دو عدد هستند که شماره‌ی انبارهای دو سر تونل i ام را نشان می‌دهند. توجه کنید که `a[0]` و `b[0]` شماره‌ی انبارهای دو سر تونل اول، و `a[n-2]` و `b[n-2]` شماره‌ی انبارهای دو سر تونل $n-۱$ اُم هستند. شماره‌ی انبارها باید عددی بین ۱ و n باشد.

تابع `init()` باید حتماً قبل از دیگر توابع کتاب‌خانه فراخوانی شود. رعایت نکردن این موضوع به منزله‌ی نقض استفاده‌ی درست از کتاب‌خانه است. برنامه‌یتان نباید با هیچ فایلی کار کند، از ورودی استانده بخواند یا در ورودی استانده چیزی بنویسد. در ضمن برنامه‌ی شما نباید سعی کند به فضای مربوط به برنامه‌ی کتاب‌خانه‌ی پیوندی دسترسی پیدا کند. بدیهی است نقض هر یک از این قوانین نمره‌ی صفر را در پی دارد و حتی ممکن است منجر به حذف شدن شما از گردانه‌ی رقابت‌ها گردد.

دو فایل `tunnels.h` و `tunnelslib.cpp` به شما داده شده است. بالای برنامه‌یتان باید عبارت `#include "tunnels.h"` را قرار دهید. تعریف توابعی که کتاب‌خانه در اختیارتان می‌گذارد، در فایل `tunnels.h` قرار دارد. برای هم‌گردانی برنامه‌ی خود فایل‌های فوق‌الذکر را در کنار برنامه (در این جا با نام `tunnels.cpp`) کپی کنید و دستور زیر را اجرا کنید:

```
g++ -O2 -static tunnels.cpp tunnelslib.cpp -lm
```

در فایل `stunnels.cpp` نیز می‌توانید نمونه‌ای از کار کردن با این کتاب‌خانه را ببینید. توجه به دو نکته ضروری است:

۱. برنامه‌ی نمونه‌ی داده شده، یعنی `stunnels.cpp` کار مفیدی انجام نمی‌دهد و تنها برای نمایش نحوه‌ی کار کردن با کتاب‌خانه فراهم شده است.

۲. در هنگام ارزیابی برنامه‌ها از کتاب‌خانه‌ی دیگری استفاده خواهد شد که توابع و تعاریف درون `tunnels.h` را پشتیبانی می‌کند.

آزمایش

کتاب‌خانه‌ی آزمایش را می‌توانید از قسمتِ Download واسط مسابقات دریافت کنید. کتاب‌خانه‌ی آزمایشیِ داده شده، داده‌های مورد نیاز را از ورودی استانده می‌خواند. باید در سطر اول ورودی، n (تعدادِ انبارها)، و در هر یک از $n-1$ سطر بعد، دو عدد بین ۱ و n نوشته شده باشند که با یک فاصله از هم جدا شده‌اند و شماره‌ی انبارهای دو سر یک تونل هستند.

هنگام استفاده از واسط آزمایش (TEST) برنامه‌یتان با همین کتاب‌خانه کار می‌کند و فایلی که به‌عنوان ورودی ارسال می‌کنید، باید دارای ساختار مذکور باشد. این کتاب‌خانه گزارش برنامه‌ی شما (ورودی تابع `report`) را بر اساس فایل ورودیتان پردازش می‌کند و درستی این گزارش را مشخص می‌کند. در صورت درستی گزارش، تعداد پرسش‌های انجام شده را نیز در خروجی استانده می‌نویسد.

شما آزادید برنامه‌هایی که دراختیارتان قرار داده شده را تغییر دهید ولی قاعدتاً نباید فایل `tunnels.h` را تغییر دهید.

شیوهٔ ارزیابی

اگر برنامه‌یتان از محدودیت‌های اعمال شده تجاوز کند، یا از کتاب‌خانه به‌صورت شایسته استفاده ننماید، به‌ازای آن موردِ آزمون، نمره‌ای به برنامه تعلق نمی‌گیرد.

برنامه‌تان حداکثر اجازه‌ی انجام ۲۰۰۰۰ پرسش را دارد. اگر بیش از این تعداد پرسش انجام دهید، برنامه‌تان متوقف شده و نمره‌ای نمی‌گیرید. اگر در پایان کار، تونل‌های گزارش شده‌ی بین انبارها نادرست باشد نیز نمره‌ای نمی‌گیرید.

برای هر مورد آزمون، یک عدد به عنوان تعداد پرسش‌های مورد نیاز برای یافتن حتمی پاسخ در نظر گرفته‌ایم. اگر این عدد $best$ و تعداد پرسش‌های شما $questions$ باشد، در صورت دادن پاسخ نهایی درست، به اندازه‌ی

$$\min\left\{۱۰۰, \max\left\{۰, ۱۲۵ - ۲۵ \times \frac{questions}{best+۱}\right\}\right\}$$

از نمره‌ی آن تست را دریافت می‌کنید.

تعامل نمونه

فراخوانی	توضیحات
`int n=init();`	مقدار `n` برابر خروجی تابع یعنی 4 می‌شود.
`bool q[4];`	تعریف آرایه‌ی از نوع bool جهت انجام پرسش‌ها.
`q[0]=q[1]=true; q[2]=q[3]=false;`	مقدار دهی `q`.
`int x=countTunnels(q);`	مقدار `x` برابر خروجی تابع یعنی 1 می‌شود.
`q[1]=q[2]=q[3]=true; q[0]=false;`	مقدار دهی `q`.
`int y=countTunnels(q);`	مقدار `y` برابر خروجی تابع یعنی 2 می‌شود.
`q[2]=q[3]=true; q[0]=q[1]=false;`	مقدار دهی `q`.
`int z=countTunnels(q);`	مقدار `z` برابر خروجی تابع یعنی 0 می‌شود.
`int a[3], b[3];`	تعریف دو آرایه از نوع int جهت دادن جواب نهایی.
`a[0]=1; a[1]=2; a[2]=4;`	مقدار دهی `a`.
`b[0]=2; b[1]=3; b[2]=2;`	مقدار دهی `b`.
`report(a, b);`	جواب نهایی را اعلام می‌کنیم.