Zadanie: POW
Poważna eksplozja

polish

ONTAK 2023, dzień 4. Dostępna pamięć: 256 MB. Limit czasu: 3 s. *03.07.2023*

Na wyspie Bajtaris wybuchł wulkan! To poważna eksplozja (nie mylić z niepoważną eksplozją), więc władze Bajtaris muszą szybko dowiedzieć się, które wioski uległy zniszczeniu, a które należy jak najszybciej ewakuować.

Wioski położone są na zboczu góry, jest ich dokładnie $2^H - 1$, i są ponumerowane od 1 do $2^H - 1$. Wioski są połączone dokładnie $2^H - 2$ szlakami. Wioska i ($i < 2^{H-1}$) jest połączona szlakiem z wioskami $2i$ oraz $2i + 1$. Im mniejszy numer wioski, tym wyżej jest ona położona.

Erupcja wulkanu nastąpiła (jak się można było spodziewać) na szczycie góry, a lawa spłynęła w dół zbocza niszcząc część wiosek na swojej drodze. Zatem, jeśli pewna wioska i została zniszczona, to wioska $\lfloor \frac{i}{2} \rfloor$ też musiała zostać zniszczona.

Władze Bajtaris mogą wysłać helikopter do zbadania terenu nad wskazaną wioską i zareportować, czy została ona zalana, czy nie. Twoim zadaniem jest zarządzać wysyłaniem helikoptera tak, aby określić, które wioski zostały zniszczone, a które nie, mieszcząc się przy tym w limicie akcji helikopterem (o tym w sekcji *Ocenianie*).

Komunikacja

To zadanie jest interaktywne. Należy napisać program, który będzie zarządzał wysyłaniem helikoptera, a następnie zaraportuje stan zniszczenia wiosek, używając do tego dostarczonej biblioteki. Należy w tym celu dołączyć nagłówek `powlib.h` za pomocą dyrektywy

```
#include "powlib.h"
```

Biblioteka udostępnia następujące funkcje:

- `int get_H()` – Zwraca parametr H ($2 \leq H \leq 17$).
- `bool query_vertex(int v)` – Wysyła helikopter nad wioskę v ($1 \leq v < 2^H$) i zwraca `true` jeśli wioska v została zniszczona i `false` w przeciwnym wypadku.
- `void answer_vertices(std::vector<bool> answers)` – Służy do zaraportowania, które wioski zostały zniszczone, a które nie. Należy zwrócić wektor rozmiaru dokładnie 2^H, gdzie w komórce i ($1 \leq i < 2^H$) powinno być `true` jeśli wioska została zniszczona i `false` w przeciwnym wypadku. Komórka 0 może zawierać dowolną wartość.

Wszelkie niezgodności ze specyfikacją komunikacji będą skutkowały złą odpowiedzią. Twój program nie powinien czytać ze standardowego wejścia, a tym bardziej starać się ingerować w działanie biblioteczki. Grozi to co najmniej złą odpowiedzią.

Biblioteka

W każdym teście to, czy wioska została zniszczona czy nie, jest ustalone z góry, więc biblioteka nie będzie dostosowywać swojego zachowania do przebiegu Twojego programu.

Ocenianie

Twoje rozwiązanie będzie oceniane na podstawie liczby użyć funkcji `query_vertex`. Oznaczmy tą liczbę przez T. Jeśli Twój program zwróci poprawną odpowiedź, nie złamie zasad interakcji z programem oraz T nie przekroczy limitu ustalonego w podzadaniu, to Twoje rozwiązanie zostanie zaakceptowane. Zestaw testów dzieli się na następujące podzadania.

Podzadanie	Ograniczenia	Punkty
1	$T \leq \frac{3}{4} \cdot 2^H, H = 2$	1
2	$T \leq \frac{3}{4} \cdot 2^H, H = 3$	8
3	$T \leq 2^H$	1
4	$T \leq \frac{7}{8} \cdot 2^H, 3 \leq H$	20
5	$T \leq \frac{3}{4} \cdot 2^H + 100$	30
6	$T \leq \frac{3}{4} \cdot 2^H$	40

Przykładowe działanie programu

W teście przykładowym zniszczone są wioski $1, 2, 3, 4, 6, 7, 12, 13$ i 15.

Wywołana funkcja	**Wynik**	**Opis**
`get_H()`	4	$H = 4$.
`query_vertex(4)`	`true`	Wioska 4 jest zniszczona.
`query_vertex(8)`	`false`	Wioska 8 nie jest zniszczona.
`answer_vertices({0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1})`	—	Odpowiadamy, że zostały zniszczone wioski $1, 2, 3, 4, 6, 7, 12, 13$ i 15. Nie pytaj, skąd ten program to wiedział.

Helikopter został wysłany dokładnie 2 razy i odpowiedź była poprawna. W teście przykładowym obowiązują ograniczenia 6. podzadania.

Eksperymenty

Przykładowe **błędne** rozwiązanie i przykładowa biblioteka znajdują się w archiwum w dziale *Pliki* na SIO_2. Biblioteka może różnić się zachowaniem od tej, która zostanie użyta do oceny rozwiązań i nie spełniać założeń zadania. Ma ona jedynie pokazać sposób interakcji z programem.

Rozwiązanie `pow.cpp` można skompilować w następujący sposób:

```
g++ -O3 -static -o pow pow.cpp powlib.cpp -std=c++17
```

Należy zadbać o to, aby pliki `powlib.h` oraz `powlib.cpp` znajdowały się w tym samym folderze co rozwiązanie.

Po skompilowaniu, biblioteka przyjmuje na standardowym wejściu opis testu, który składa się z dwóch wierszy. Pierwszy z nich powinien zawierać pojedynczą liczbę H. Drugi z nich powinien zwierać $2^H - 1$ liczb 0 lub 1, gdzie i-ta z nich oznacza odpowiednio, że wioska i ocalała lub została zniszczona. Plik wejściowy odpowiadający testowi przykładowemu znajduje sie w archiwum w pliku `pow0.in`.

Завдання: POW
Poważna eksplozja

ukrainian

ONTAK 2023, день 4. Обмеження пам'яті: 256 MB. Ліміт часу: 3 s. *03.07.2023*

Опаааааа, боєприпаси зі збідненим ураном працюють як зі звичайним!

По Байтгородській Народній Республіці запустили перший НАТОвський боєприпас зі збідненим ураном. Удар був по військовому табору ворога, і тепер нашій розвідці треба дізнатися, які казарми вже знищені, а по яким відповідно буде наступний залп.

Казарми розташовані на схилах гори, їх рівно $2^H - 1$, і вони пронумеровані від 1 до $2^H - 1$. Казарми з'єднані рівно $2^H - 2$ трасами. Казарма i ($i < 2^{H-1}$) з'єднана трасою з казармами $2i$ та $2i + 1$. Чим менший номер казарми, тим вище вона розташована.

Попадання снаряду відбулося (як це було очікувано) по самій вершині гори, і ядерна маса стекла вниз по схилах, знищуючи частину казарм на своєму шляху. Тому, якщо певна казарма i була знищена, то казарма $\lfloor \frac{i}{2} \rfloor$ також мусила бути знищеною.

Наша розвідка може відправити гелікоптер (як вона вже робила не один раз) для огляду території над вказаною казармою і звітувати, чи була вона затоплена, чи ні. Ваше завдання - керувати відправленням гелікоптера так, щоб визначити, які казарми були знищені, а які ні (поки що), дотримуючись при цьому ліміту дій гелікоптера (про це в розділі *Оцінювання*).

Комунікація

Це завдання є інтерактивним. Ви повинні написати програму, яка буде керувати відправленням гелікоптера, а потім звітувати стан зруйнування казарм, використовуючи для цього надану бібліотеку. Ви повинні в цій меті включити заголовок `powlib.h` за допомогою директиви

```
#include "powlib.h"
```

Бібліотека надає наступні функції:

- `int get_H()` – Повертає параметр H ($2 \leq H \leq 17$).
- `bool query_vertex(int v)` – Відправляє гелікоптер над казармою v ($1 \leq v < 2^H$) і повертає `true` якщо казарма v було знищено і `false` в протилежному випадку.
- `void answer_vertices(std::vector<bool> answers)` – Використовується для звітування, які казарми були знищені, а які ні. Потрібно повернути вектор розміру рівно 2^H, де в комірці i ($1 \leq i < 2^H$) повинно бути `true` якщо казарма була знищена і `false` в протилежному випадку. Комірка 0 може містити будь-яке значення.

Будь-які невідповідності до специфікації комунікації будуть мати наслідком неправильну відповідь. Ваша програма не повинна читати зі стандартного вводу, а тим більше намагатися втрутитися в роботу бібліотеки. Це загрожує принаймні неправильною відповіддю.

Бібліотека

У кожному тесті те, чи казарма була знищена чи ні, визначається наперед, і бібліотека не буде налаштовувати свою поведінку по ходу вашої програми.

Оцінювання

Ваше рішення буде оцінюватися на основі кількості використань функції `query_vertex`. Позначимо цю кількість через T. Якщо ваша програма поверне правильну відповідь, не порушить правила взаємодії з програмою та T не перевищить ліміт, встановлений у підзадачі, то ваше рішення буде прийнято. Набір тестів ділиться на наступні підзадачі.

Підзадача	Обмеження	Бали
1	$T \leq \frac{3}{4} \cdot 2^H, H = 2$	1
2	$T \leq \frac{3}{4} \cdot 2^H, H = 3$	8
3	$T \leq 2^H$	1
4	$T \leq \frac{7}{8} \cdot 2^H, 3 \leq H$	20
5	$T \leq \frac{3}{4} \cdot 2^H + 100$	30
6	$T \leq \frac{3}{4} \cdot 2^H$	40

Приклад роботи програми

У прикладовому тесті знищені казарми $1, 2, 3, 4, 6, 7, 12, 13$ та 15.

Викликана функція	**Результат**	**Опис**
`get_H()`	4	$H = 4$.
`query_vertex(4)`	`true`	Казарма 4 знищена.
`query_vertex(8)`	`false`	Казарма 8 не знищена.
`answer_vertices({0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1})`	—	Відповідаємо, що були знищені казарми $1, 2, 3, 4, 6, 7, 12, 13$ і 15. Не питайте, звідки ця програма це знала.

Гелікоптер було відправлено рівно 2 рази і відповідь була правильна. У прикладовому тесті діють обмеження 6. підзадачі.

Експерименти

Прикладове **неправильне** рішення і прикладова бібліотека знаходяться в архіві в розділі *Файли* на SIO_2. Поведінка бібліотеки може відрізнятися від тієї, яка буде використана для оцінювання рішень іможе не виконувати припущень завдання. Вона лише має показати спосіб взаємодії з програмою.

Рішення `pow.cpp` можна скомпілювати наступним чином:

```
g++ -O3 -static -o pow pow.cpp powlib.cpp -std=c++17
```

Вам слід подбати про те, щоб файли `powlib.h` та `powlib.cpp` знаходилися в тій же директорії, що і ваше рішення.

Після компіляції, бібліотека приймає на стандартному вводі опис тесту, який складається з двох рядків. Перший з них повинен містити одне число H. Другий з них повинен містити $2^H - 1$ чисел 0 або 1, де i-те з них означає відповідно, що казарма i вижила або була знищена. Вхідний файл, що відповідає прикладовому тесту, знаходиться в архіві у файлі `pow0.in`.