КОНТРОЛНО СЪСТЕЗАНИЕ
НА РАЗШИРЕНИЯ НАЦИОНАЛЕН ОТБОР
2-4 юни 2023 г.
Група А, 11-12 клас

Задача А3: Роботи 5?

Задачата е проста – даден ненасочен граф с N върха и M ребра; проверете дали е свързан. Разбира се, това не е всичко, тъй като програмата, която трябва да напишете, следва специален протокол. Тя ще се изпълнява постъпково от N робота, по един във всеки връх, които разполагат със специален вид памет. На всяка стъпка всеки робот ще изпълнява Вашата функция `work`. Това ще се повтаря докато програмата не приключи изпълнение.

Памет:

Разполагате с два вида памет:

1. "Глобална памет" – всеки робот може да чете и пише в нея.
2. "Върхова памет" – специфична за всеки връх, съдържа съседите му и може да бъде четена само от робота в него. Тя остава еднаква през цялото изпълнение на програмата.

Тази глобална памет е предоставена от журито. Повече детайли за имплементацията ще видите по-долу.

Роботи:

Разполагате с N робота, като робот номер u се намира във връх номер u за $1 \leq u \leq N$. Протоколът е базиран на T стъпки, като на всяка стъпка, всеки робот трябва да направи следното:

1. Свободно чете от "глобалната памет" от стъпка T и "върховата памет" на върха, в който се намира (формално робот u може да чете съседите на връх u, като това е налично на всяка стъпка). Няма ограничение за броя прочетени клетки от паметта (стига програмта да не наруши тайм лимита).
2. Прави **точно 5** писания върху глобалната памет, **като тези писания ще бъдат видяни от другите роботи и себе си на стъпка** $(T + 1)$.

Роботите се изпълняват *атомично*. Това означава, че всеки робот изпълнява програмата без да бъде прекъснат от другите роботи. Редът на изпълнение на роботите за всяка стъпка винаги е случаен. Повече детайли може да видите по-долу в параграфа за детайли по имплементацията.

Глобална памет:

Глобалната памет представлява $5N + 3$ клетки номерирани $0, \ldots, 5N + 2$ (напомняме че N е броя върхове в графа). Всяка от тях пази цяла стойност между 0 и $2^{32} - 1$ (включително). Първоначално (глобалната памет за стъпка $T = 1$), всяка клетка от паметта съдържа стойност 0, освен 2 специални клетки:

- Клетка номер 1 съдържа броя върхове N.
- Клетка номер 2 съдържа броя ребра M.

Клетка номер 0 също е специална, като тя е запазена за **вашия отговор**. Ако след стъпка T, вашата програма съдържа стойност $x \neq 0$ в клетка 0, програмата терминира. Ако $x = 1$, прогрмата заявява, че графът е свързан. Ако $x = 2$, програмата заявява, че графът е **не е** свързан. Ако клетката още съдържа 0 се преминава към стъпка $T + 1$.

КОНТРОЛНО СЪСТЕЗАНИЕ
НА РАЗШИРЕНИЯ НАЦИОНАЛЕН ОТБОР
2-4 юни 2023 г.
Група А, 11-12 клас

Ограничения

- $2 \leq N \leq 50000$

Подзадачи и оценяване

Подзадача	Точки	Допълнителни ограничения
1	**19**	$N \leq 50$
2	**21**	$N \leq 250$
3	**25**	$N \leq 1000$
4	**3**	$N \leq 50000$ и графът не съдържа цикли
5	**32**	$N \leq 50000$

Всяка подзадача съдържа няколко теста, като ако вашата програма не намери правилно дали графа е свързан, ще получите 0 точки. Също така, ще получите 0 точки ако протоколът не терминира след 200000 стъпки. Частта от точките, които ще получите на дадена подзадача зависи от максималния брой стъпки T, които правите на подтест преди да терминирате протокола:

$$\min\left(1.0, \sqrt{\frac{2\lceil \log_2(N) \rceil}{T}}\right)$$

Детайли по имплементацията

Вашият код ще стои в хедър файл, който ще бъде включен в "средата" на кода на журито. Материалите включват примерен хедър `solution.h`, който съдържа примери за как точно се имплементира протокола. Хедъра трябва да включва следнтие дефиниции:

```
#define SET_VAL 0
#define ADD_VAL 1

#define ANSWER_CONNECTED 1
#define ANSWER_NOT_CONNECTED 2

struct MemoryOperation {
    bool type;
    unsigned int addr, val;
    constexpr MemoryOperation(
        bool type,
        unsigned int addr,
        unsigned int val
    ) : type(type), addr(addr), val(val) {}
};
```

КОНТРОЛНО СЪСТЕЗАНИЕ
НА РАЗШИРЕНИЯ НАЦИОНАЛЕН ОТБОР
2-4 юни 2023 г.
Група А, 11-12 клас

```
constexpr std::array<MemoryOperation, 5> work(
    unsigned int u,
    unsigned int *memory,
    unsigned int memory_size,
    unsigned int *neighbours,
    unsigned int number_neighbours
);
```

Вие трябва да имплементирате само `constexpr` функцията `work`, като тя представлява кода на всеки робот. Тази функция приема като аргументи:

- `u` - върха в който се намира съответния робот.
- `memory` - "глобалната памет", дадена като масив от `unsigned int`.
- `memory_size` - броя клетки в глобалната памет. Тази стойност винаги е равна на $5N + 3$ (където N е броя върхове).
- `neighbours` - "върховата памет", или масив съдържащ съседните върхове на `u`.
- `number_neighbours` - броя на съседните върхове на `u` (както и големината на масива `neighbours`).

Функцията трябва да върне масив с **точно 5** елемента, като всеки е от зададената по-горе структура `MemoryOperation`. `MemoryOperation` представлява операция за писане върху паметта, като тя може да е от 2 вида.

- `MemoryOperation(SET_VAL, addr, val)` – глобалната памет в адрес `addr` на **следващата стъпка** ще бъде презаписана със стойност `val`, или:

  ```
  memory[addr] = val
  ```

- `MemoryOperation(ADD_VAL, addr, val)` – към глобалната памет в адрес `addr` на **следващата стъпка** ще бъде добавена стойност `val`, или:

  ```
  memory[addr] += val
  ```

При конфликт, т.е. ако два или повече робота искат да изпълнят операция върху една и съща клетка, операциите се изпълняват първо в реда на роботите (който е произволен на всяка стъпка), а после в реда, в който функцията `work` ги връща. Също така, използвайки `ADD_VAL` е позволено стойностите да overflow-ват (т.е. те винаги са по модул 2^{32}).

На всяка стъпка T, функцията `work` ще бъде извикана точно веднъж за всеки робот. Редът на извикване на функцията ще бъде случаен и не е гарантирано, че ще е еднакъв за всяка стъпка. Може да използвате локалния грейдър `Lgrader.cpp` за да видите допълнителни детайли по протокола.

Локалният грейдърът чете от първия ред N, и S, където S е сийд за генериране на случайния ред на роботите на всяка стъпка. От следващите M реда се въвеждат и ребрата на самия граф (по две числа на ред). Грейдъра включва хедър файла който трябва да имплементирата, така че за тестване стига да компилирате само грейдъра.

Системният грейдър **не е** адаптивен.

КОНТРОЛНО СЪСТЕЗАНИЕ
НА РАЗШИРЕНИЯ НАЦИОНАЛЕН ОТБОР
2-4 юни 2023 г.
Група А, 11-12 клас

Детайли за `constexpr`

Спецификацията `constexpr` за функция означава, че тя няма (или по-точно може да няма) странични ефекти, т.е. няма да използва глобални не-`constexpr` променливи и фунцкии, няма да въвежда или извежда и т.н. Може да използва локални променливи или (потенциално Ваши) глобални `constexpr` константни, стига те да са инициализирани при създаването си, също може да вика други (потенциално Ваши) `constexpr` функции, да има `if`-ове и цикли и всички други такива прости конструкции.

Тъй като C++ не задължава `constexpr` функциите да са винаги `constexpr`, програмата на журито ще съдържа един вграден тест, на който програмата Ви ще бъде "тествана" по време на компилация и няма да се компилриа успешно, ако не е `constexpr` на него. Забележете, че е възможно умишлено да напишете програма, която се компилира, но не е винаги `constexpr`, т.е. понякога използва глобални променливи. При откриване на такова решение, то ще се анулира, а при повторни нарушения, ще бъдете дисквалифицирани. Тъй като програмата на журито, ефективно пуска Вашата програма на пълен тест по време на компилацията е възможно (но много трудно) самата компилация да има ML (или TL). Това би се случило, ако програмата Ви и без това би имала TL или ML или има някакви много дълбоки рекурсии.