НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
НАЦИОНАЛЕН КРЪГ, 12-14 март 2021 г.
Група АБ, 9 – 12 клас

Задача АВ3. Стекланг

Фриц си построил една машинка. Тя наподобява модерен компютър, но цялата и памет се състои само от различни стекове от цели числа. Стек е структура, която поддържа списък от елементи и в даден момент дава достъп само до най-горния/първия елемент. Този елемент може да бъде четен, модифициран, махан или друг елемент може да бъде добавен върху/пред него. Махане на първия елемент на даден стек ще наричаме „попване“, а добавяне на нов елемент най-отпред ще наричаме „пушване“.

С цел да управлява новата си машинка, Фриц е създал програмен език за нея, а именно Стекланг. Всяка програма на Стекланг се състои от списък от инструкции, които имат по един или повече параметри. Например инструкцията `pop S` попва от стека `S`, а инструкцията `push S 10` пушва числото `10` в `S`. Също така има инструкции за аритметични операции, control flow (if и goto) и IO (вход и изход). Пълен списък с инструкции ще бъде предоставен по-долу в условието. Инструкциите се изпълняват една след друга, от горе надолу, освен ако не се изпълни control flow инструкция. Езикът също така поддържа елементарни коментари, с цел по-лесно програмиране на него.

Сега Фриц иска да вкара машинката си в употреба, като я използва, за да решава разни трудни задачи. Първата такава, на която се е спрял, е доста стандартна в света на компютърните науки: да се намери най-късия път между два върха в насочен граф, чиито ребра имат тегла или 0 или 1. Върховете можем да си представим като градове, ребрата с тегло 1 – като еднопосочни пътища, а ребрата с тегло 0 – като еднопосочни портали. Фриц иска да намери теглото (сумата на теглата на всички ребра по пътя) на най-лекия път от даден начален връх до даден краен връх, ако въобще има такъв път. С цел да улесни входа, той е добавил няколко входни функции към Стекланг специално за тази задача.

Фриц обаче е изключително изморен от строенето на машинката, а и от създаването на самия език. Помогнете му, като напишете програма `stacklang.txt` на Стекланг, която да намира теглото на най-лекия път в зададен граф или да връща -1, ако няма такъв.

Локално тестване

С цел да тествате решението си, Ви е предоставен интерпретатор на езика Стекланг. На него му се задава граф и програма. След това той „компилира“ програмата, изпълнява я върху зададения граф и отпечатва върнатия резултат (или съобщение за грешка).

Вход на интерпретатора

От първия ред на входа се въвеждат две числа: N и M – броя върхове и броя ребра. От втория ред се въвеждат $Start$ и End – началния и крайния връх на пътя. На всеки от следващите M реда се въвеждат по три числа: $From_i$, To_i и Len_i – началния и крайния връх на i-тото ребро, както и теглото му. След това се въвежда програмата във формата описан по-долу.

Формат на програмата

Форматирането на програмата няма значение (може да има индентации и прочие), но трябва всички токъни (инструкции, стекове, разделители за коментари и т.н.) да са разделени един от друг с интервал, нов ред или друг white space символ. Коментарите се ограждат със символа `#`. Например: `code # comment # code`. Програмата се чете до край на файл, който може да „напишете“ в конзолата с Ctrl+Z на Windows.

НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
НАЦИОНАЛЕН КРЪГ, 12-14 март 2021 г.
Група АБ, 9 – 12 клас

Инструкции

За да дефинираме Стекланг точно, разглеждаме няколко типа неща: I е инструкция, C е условие, Z – цяло число, S – стек, L – маркер, а R – или цяло число Z, или стек S. Също така дефинираме $top(S)$ като най-горния елемент на S, а $top(Z)$ като просто Z.

Тип	Възможност	Описание
I	`alloc` Z S_1 S_2 … S_Z	Алокира (заделя памет за) Z стека с имена S_1, S_2, …, S_Z.
I	`pop` S	Попва от S.
I	`push` S R	Пушва $top(R)$ в S.
I	`add` S R	$top(S) := top(S) + top(R)$
I	`sub` S R	$top(S) := top(S) - top(R)$
I	`print` R	При вас отпечатва $top(R)$. Не прави нищо на системата.
I	`return` R	Връща $top(R)$, т.е. това е резултата на програмата.
I	`getStart` S	Пушва началния връх в S.
I	`getEnd` S	Пушва крайния връх в S.
I	`getEdges` S R	Пушва всички ребра* от връх $top(R)$ в S.
I	`label` L	Дефинира маркер L на това място в кода.
I	`goto` L	Изпълнението на програмата скача към L.
I	`if` C `goto` L	Изпълнението на програмата скача към L ако C е вярно.
C	`empty` S	Вярно ако S е празен.
C	`notEmpty` S	Вярно ако S не празен.
C	`equal` R_1 R_2	Вярно ако $top(R_1) = top(R_2)$.
C	`notEqual` R_1 R_2	Вярно ако $top(R_1) \neq top(R_2)$.

ВАЖНО: **Всички `get` инструкции пушват желаните неща само при първото си извикване.** Всички следващи извиквания не правят нищо, т.е. например второ извикване на `getEdges` с едно и също $top(R)$ няма да пушне нищо в S, но със различно $top(R)$ – ще.

* Редът на данните за ребрата е такъв, че след извикването на `getEdges` всички ребра с тегло 1 ще са преди всички ребра с тегло 0, а за дадено ребро първо стои теглото му и после върхът, до който води то. Т.е. отгоре на S ще има: $1, v_1, \dots, 1, v_t, 0, v_{t+1}, \dots, 0, v_{t+p}$, където v_i са всички съседи на връх $top(R)$, от които t са през ребро с тегло 1, а p – с тегло 0.

НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
НАЦИОНАЛЕН КРЪГ, 12-14 март 2021 г.
Група АБ, 9 – 12 клас

Забележки:

- Изпълнение на инструкция, която използва $top(S)$ за празен стек S е грешка.
- Изпълнение на `getEdges` за невалиден връх е грешка.
- Използване на недефиниран (от `alloc` или `label`) стек или маркер е грешка.
- Диапазонът на числата в стековете е същият като на `int` в C++. При излизане от диапазона, поведението е същото както в C++.
- Валидни имена на стекове и маркери са същите както на променливи в C++ и не трябва да съвпадат с име на инструкция.
- Числата (Z или R) могат да бъдат и отрицателни.

Подзадачи и оценяване

За да получите точки за дадена подзадача, решението Ви трябва успешно да премине всички тестове в нея, а за да премине даден тест, решението Ви трябва да не произведе никакви грешки, да алокира не повече от 10 стека и да върне верен отговор в до 10^7 итерации. Процента от точките, които решението Ви ще вземе за дадена подзадача, зависи от броя стекове, които то алокира, и потенциално от максималния брой итерации, които то е използвало на който и да е тест (всъщност подтест) в подзадачата.

Подзадача	Точки	Ограничение
1	20	Всички ребра са с тегло 0.
2	20	От всеки връх има не повече от един път до всеки друг.
3	30	Всички ребра са с тегло 1.
4	30	Няма.

Стекове	Итерации	Процент точки
≤ 2	$\leq 3 \times 10^6$	100%
≤ 2	$> 3 \times 10^6$	90%
3	Без значение	70%
4	Без значение	50%
5	Без значение	40%
≥ 6	Без значение	30%

Забележете, че тъй като за финалния Ви резултат на задачата се гледа най-добрия Ви събмит за всяка подзадача поотделно, има смисъл, да кажем, да качите решение с 4 стека за цялата задача, но такова с 3 стека специално за някоя конкретна подзадача.

Забележка: Всеки тест на системата всъщност ще включва по няколко подтеста. Решението Ви ще бъде пускано на всеки подтест поотделно и ще премине теста успешно, само ако премине всички подтестове. Това по никакъв начин не влияе на решението Ви.

НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
НАЦИОНАЛЕН КРЪГ, 12-14 март 2021 г.
Група АБ, 9 – 12 клас

Ограничения за графа

$1 \leq N \leq 2 \times 10^4$
$0 \leq M \leq 5 \times 10^4$
$0 \leq Start, End, From_i, To_i < N$
$0 \leq W_i \leq 1$

Забележка: В графа може да има ребра от връх до себе си, може да има над едно еднакви ребра и може началният и крайният връх да са равни.

Примерни графове

Граф 1	Отговор 1	Граф 2	Отговор 2
4 7 0 2 0 1 1 1 1 1 0 3 0 2 0 0 0 3 1 3 1 0 1 2 1	1	3 2 1 0 0 2 1 2 1 0	-1

Най-лекия път от 0 до 2 в първия граф е от 0 до 3 с тегло 0, от 3 до 1 с тегло 0 и накрая от 1 до 2 с тегло 1. Във втория граф няма път от 1 до 0.

Примерна програма

```
alloc 4 start end edges cnt
getStart start
getEnd end

push cnt 0
getEdges edges start
getEdges edges end

label edgesLoop
if empty edges goto exitEdgesLoop
  add cnt 1
  pop edges # don't need len #
  print edges
  pop edges
goto edgesLoop
label exitEdgesLoop

# cnt needs to be doubled when
  start and end are equal #
if notEqual start end goto finish
  add cnt cnt

label finish
return cnt
```

(Дадена е в две колони за да се събере на страницата.)

Тази програма не решава възложената Ви задача. Вместо това, тя намира броя ребра, излизащи от $Start$, плюс броя ребра, излизащи от End. Забележете възможността да е нужно удвояване на брояча (`add cnt cnt`), когато $Start = End$. Това е нужно, защото **тогава `getEdges edges end` няма да пушне нищо** и затова трябва ръчно да преброим всяко ребро по два пъти като удвоим брояча накрая. Програмата също така печата всеки връх, до който има ребро от $Start$ или End. Това печатане не би се изпълнило на системата.