Анализ на решението на задача
Локална мрежа

За обяснение на решението ще използваме примера от условието на задачата, а именно:

Вход:	**Изход:**
5 4 4 4 4 1 3 3 5 2 5	10110 0110

CU1 има 5 порта
CU2 има 4 порта

Установяваме всички портове на **CU1** в режим на получаване и всички портове на **CU2** в режим на изпращане. Веднага забелязваме, че някои от получаващите портове на **CU1** бездействат - няма портове от **CU2**, които да им изпращат информация. Ще продължим да поддържаме това състояние -> след всяка стъпка решението ще бъде ОК, с изключение на това, че някои получаващи портове на **CU1** ще продължават да бездействат.
Последователно избираме приемащ порт на **CU1**, който не получава информация. Ако няма такива портове, ние сме готови. Нека **`X`** да е избраният порт. Превключваме порт **`X`** в режим на изпращане. Нека порт **`Y`** от **CU2** е дестинацията на **`X`**. Ако порт **`Y`** е в режим на получаване, всичко е наред – запазваме статуквото. В противен случай просто превключваме порт **`Y`** в режим на получаване. По този начин можем да направим друг получаващ порт на **CU1** "Бездействащ", но статуквото все още е вярно.
Накрая няма останали бездействащи портове, така че сме постигнали правилно решение.
На всяка стъпка броят на приемащите портове на **CU1** намалява с 1, така че ще направим най-много **`M`** стъпки. Толкова, колкото е броя на портовете на **CU1.**

За ускоряване на алгоритъма, въвеждаме масив **`inDegree`**, в който съхраняваме за всеки порт на **CU1** броя на портовете на **CU2**, които изпращат до него. Освен това следим за всички бездействащи, приемащи портове на **CU1** - те се съхраняват в стек **`zeroDeg`**. На всяка стъпка вземаме порт от стека, превключваме го в режим на изпращане, възможно е да променим режима на приемащия порт и да актуализираме един запис в масива inDegree.

Четенето на входа и началната стъпка отнемат време O (m + n), поради което времевата сложност на целия алгоритъм е O (m + n). Нуждаем се също от пространство O (m + n).

Друго решение.
Бездействащ порт може да се търси и без използване на масив **`inDegree`**. Тогава времето за изпълняване на всяка стъпка нараства. Файлът setp2.cpp съдържа програмна реализация на това решение.

Всички тестове са генерирани автоматично и са случайни. Някои тестове, които имат единствен отговор са с висока трудоемкост, защото се увеличава времето за претърсване, при определянето на нужния порт.

Задачата може да има повече от един решение, затова за проверката е необходим чекер: Файл - setpcheck.cpp. Намира се във фолдер author.

Автор Пано Панов