Подзадача 1 $1 \leq N \leq 10; 1 \leq S, T \leq 5$ – **10 т.**

За тази подзадача се очаква да мине всяко наивно решение например такова със сложност $O(2^N N^2)$. Пробваме всички комбинации ученици (2^N) и за всяка наивно проверяваме дали няма повторения в уменията им (N^2). Така намираме максималния размер на отбора, както и кои ученици и умения някога/винаги/никога са включени в него.

Подзадача 2 $1 \leq N \leq 6 \times 10^2; 1 \leq S, T \leq 3 \times 10^2$ – **30 т.**

За тази подзадача трябва вече да разберем, че за задачата всъщност се говори за максимален мачинг в двуделен граф. За намиране на максималния размер на отбора, можем да използваме какъв да е алгоритъм за мачинг. За сега ще се придържаме към преобразуване към поток и след това използване на алгоритъм за макс флоу. Тук е предвиден алгоритъм на Форд-Фулкерсон. Интересната част на задачата е как да открием кои върхове и ребра присъстват във всички/никои от мачингите.

Ще използваме стандартен трик за всякакви такива проблеми. За всеки елемент (връх или ребро) ще пробваме да го махнем и да намерим макс мачинга без него. Ако без него е отговорът е по нисък, значи елемента е присъства във всички максимални мачинги. Подобна е ситуацията и на обратно. Ако накараме ребро да е в мачинга и той се понижи, значи то никога не може да присъства. Можем да накараме ребро да присъства в мачинга, като всъщност от графа махнем него и върховете, които то свързва. След това пускаме алгоритъма и накрая (фиктивно) отново добавяме върховете и реброто и ги свързваме.

Оставащия проблем е как да накараме върхове да са вътре в графа. Това е по трудно, защото за разлика от ребрата те може да се мачнат с много други върхове и би ни трябвало да пробваме с всеки. За щастие можем да направим следното наблюдение. Да кажем, че връх А не присъства в макс мачинга, който сме открили, и той има ребро към връх Б. Ако Б също не присъства в макс мачинга, можем да ги мачнем и да увеличим мачинга, което е противоречие Ако Б присъства в мачинга и мачнат към В, можем да ънмачнем Б от В и да мачнем А с Б. Това запазва макс мачинга и вече А е мачнат. Следва, че връх не присъства в нито един макс мачинг тогава и само тогава, когато е от степен 0.

С това покриваме всички четири групи. Сложността на решението е $O(N \times N^2)$. От това $O(N^2)$ е за потока, което всеки опитен състезател трябва има предвид, защото на практика потоците често се държат доста по бързо от сложността си.

Подзадача 3 $1 \leq N \leq 1{,}5 \times 10^3; 1 \leq S, T \leq 7{,}5 \times 10^2$ – **40 т.**

Тук предвиденото решение е доста подобно на това за предната подзадача. Направени са две подобрения, от които едното е по съществено. Първото (по-малко важно) е, че в предното решение ще проверим дали ребро (или връх) винаги е нужно, дори когато то не присъства в оригиналния ни макс мачинг. Тук всеки пък като направим различен макс мачинг, обхождаме всички елементи и записваме дали все още е валидни хипотезите „този елемент винаги е нужен“ и „този елемент никога не е нужен“. Това гарантира, че за даден елемент ще тестваме най-много една от двете хипотези (а понякога и нито едната). Използват се и други подобни наблюдения, като това, че ако ребро е винаги нужно, то и съответните му върхове винаги са нужни, и че ако връх не е нужен винаги, то всяко негово ребро може да се използва. Второто следва от доказателството в края на предната част.

Второто подобрение е да използваме по-бърз алгоритъм за максимален поток, а именно алгоритъм на Диниц. Така сложността става $O(N \times N\sqrt{N})$, а и с по-добра константа. Всъщност за вземането на подзадачата не са нужни всички наблюдения.

Подзадача 4 $1 \leq N \leq 4 \times 10^3; 1 \leq S, T \leq 2 \times 10^3$ **– 55 т.**

Тук все още се придържаме към същия стил решение, но коригираме един проблем. В предните две решения всеки път като тестваме някоя хипотеза пускаме нов поток изцяло от нулата. Това очевидно е доста неефикасно. Затова ще поддържаме остатъчния граф винаги да е в състояние на максимален поток. Като искаме да тестваме някоя хипотеза, първо ще изпразним съответното ребро или връх, след това ще го махнем и накрая ще пуснем само едно DFS за възстановяване на потока (ако е възможно). След това можем да върнем въпросните елементи в графа и след като направим проверката, ако потока се е намалил, отново да пуснем функцията за възстановяване на поток. Като идея това не е много сложно, но имплементацията си има своите особености.

Сложността става $O(N^2)$. Забележете, че вътрешното DFS, макар и за поток, си се държи изцяло линейно на някои видове тестове. Това е защото това е последното DFS за потока (за последната единица поток) и на лош тест това може да изисква обхождане на голяма част от графа. Възможни са различни евристики и оптимизации, които да подобрят това, но не са нужни за преминаване на тази подзадача.

Рандомизирани решения

Преди да преминем към пълните решения е добре да разгледаме още един вид решение. Можем да се опитаме да приложим подобен похват както оптимизациите за подзадача 3, но без същинската част от решението. Т.е. само да пускаме произволни максимални потоци и за всеки елемент да следим дали някога/винаги/никога се е срещал. Накрая след фиксирана бройка максимални потоци (или докато не ни изтече времето), приемаме, че данните ни са репрезентативни за всички възможности.

Със средно очевидни наблюдения можем да видим, че ако даден елемент се използва някога, той се използва в поне $1/N$-та от всички максимални мачинги. Т.е. няма да се наложи да правим експоненциална бройка произволни решения за да видим всичко възможно. Също така отделните елементи не са независими, което ни помага. В крайна сметка, за фиксирана вероятност за успех p са ни нужни линейна по N бройка произволни максимални мачинги. За това сложностите на такива решения за константно p е отново като на горните, т.е. са потенциално възможни за 30 до 55 точки.

Забележете, че тук също може да не пускаме изцяло нови потоци винаги а само да сменяме някои неща вътрешно, което да забърза решението (т.е. да пуска произволни нулеви потоци по разни цикли).

Подзадача 5 $1 \leq N \leq 1.8 \times 10^4; 1 \leq S, T \leq 9 \times 10^3$ **– 75 т.**

Тук вече навлизаме в същинското решение на задачата. До идеите за това решение можем да стигнем по няколко начина. Ако сме си мислели за решение като това на подзадача 4, можем да се чудим дали няма някакъв по бърз начин да правим проверката за даден елемент. Ако сме си мислели за някакво рандомизирано решение, може да се чудим как да пускаме такива нулеви потоци по цикли и дали не можем да гледаме това директно. Естествено е възможно и директно да тръгнем в тази насока.

Идеята тук, както вече беше загатнато, е да гледаме задачата конкретно в контекста на потоци. След като направим един макс мачинг искаме да видим в линейно време кои са всички ребра и върхове, които могат да сменят състоянието си (от използвани към неизползвани и обратното). След това за всяка категория тези, които не могат да сменят състоянието си и присъстват в оригиналния мачинг, са винаги нужни и подобно за никога.

Едно много удобно наблюдение е, че в контекста на максимален поток за всеки връх в графа има точно едно ребро, което се използва тогава и само тогава, когато върхът се използва. За левия дял това е реброто от фиктивния сорс до върха, а за десния дял – реброто от върха до синка. Така всъщност пълното решение е доста чисто, то разглежда само ребра във флоу граф и после само декодираме тези данни, за да отговорим на задачата. Всъщност решението решава задачата (за ребра) в произволен флоу граф с единични капацитети на ребрата.

Нека разгледаме остатъчните графове на два различни максимални потока. Щом са максимални, значи няма път от сорса до синка. Въпросът е как от единия остатъчен граф да получим другия (или по-точно всеки друг). Ами единствените валидни потоци (такива, които да уважават свойствата на всеки поток) са цикли в остатъчния граф. Това е, защото всеки не цикличен път ще наруши свойството за запазване на поток на крайните си два върха, а единствените два върха, за които това е позволено, са сорсът и синкът, но както вече казахме такъв път няма. Т.е. пускане на поток по произволен цикъл в остатъчен граф на един максимален поток, води до остатъчен граф на друг максимален поток. Твърдението е вярно и в обратната посока. Това може да се докаже сравнително лесно по противоречие, но тук ще пропуснем доказателството.

Сега нека разгледаме какъв ефект има това пускане на поток по цикъл върху състоянията на ребрата по него. За тези, които се използват в момента, ребрата, по които можем да пуснем поток, всъщност са техните обратни, а за тези, които не се използват в момента, това са самите те. Като пуснем потока, това ще запълни капацитетите на всички ребра по цикъла и съответно ще обърне състоянието им (и ще увеличи капацитета на техните обратни ребра). Т.е. ако дадено ребро участва в който и да е цикъл в остатъчен граф на който и да е максимален поток, то може да сменя състоянието си (нито се използва винаги, нито никога). От друга страна, ако не участва в цикъл в остатъчния граф на даден максимален поток, то ще е в това състояние във всеки максимален поток.

Остава да намерим кои ребра участват в цикъл. Това става лесно като намерим силно свързаните компоненти на остатъчния граф. Дадено ребро участва в цикъл, ако свързва два върха от една и съща силно свързана компонента.

Решението се състои от едно намиране на максимален поток плюс няколко линейни стъпки. Общата му сложност е тази на алгоритъма за максимален поток. За тази подзадача е предвидено да мине всеки такъв алгоритъм, например алгоритъм на Форд-Фулкерсон. Т.е. сложността е $O(N^2)$, но това е доста по бързо от за предната подзадача, защото реално не правим N^2 стъпки, тъй като това е поток (първите пътища се откриват много бързо).

Подзадача 6 $1 \leq N \leq 1{,}5 \times 10^5; 1 \leq S, T \leq 7{,}5 \times 10^4$**– 100 т.**

Тук същината на решението не се променя. Просто използваме по-бърз алгоритъм за максимален поток. Използвания в едно от приложените решения е алгоритъм на Диниц. Това има сложност $O(N\sqrt{N})$. Друга опция е да използваме някой силно оптимизиран алгоритъм директно за мачинг (чиято сложност на теория е по-лоша) и след това да пренесем неговите резултати върху остатъчни граф. Възможно е и да не пускаме втората част (тази със силно свързаните компоненти) експлицитно в остатъчния граф (а може и да нямаме такъв, ако не сме използвали максимален поток за първата стъпка), а да я правим директно върху мачинга. В крайна сметка резултатът е същият и отново търсим силно свързаните компоненти. Имплементации на тези различни варианти може да намерите в приложените решения.