**Анализ**

Задачата задава типична графова постановка – градовете са върхове на графа, директните магистрални отсечки са неориентирани ребра между тях с тегла – времето на изминаване. Понятията, които са въведени в условията са икономичен маршрут, който е маршрут с минимална дължина, ако разглеждаме графа като непретеглен, и най-бърз маршрут, който е такъв с минимално тегло в претегления граф. Решението комбинира поредица от стандартни алгоритми след като се направят ключови наблюдения.

Първата подзадача е за 21 точки. Идеята е сравнително тривиална, но все пак изисква по-сложен алгоритъм. Тук имаме улеснения за тестовете – най-бързият маршрут между началния и крайния град е и икономичен. Това означава, че може да не мислим все още за понятието икономичен маршрут. Достатъчно е да намерим най-бързия маршрут между градове 1 и ***N*** и той ще е търсеният отговор. Такъв маршрут може да намерим, като напишем алгоритъма на Дийкстра. Така сложността за тази подзадача е .

Втората подзадача е за 10 точки. Тук вече нямаме улеснението от миналата подзадача. Търсим най-бързият икономичен маршрут. Това означава, че първо търсим да е с минимален брой изминати върхове, а от тези маршрути, този с най-малко време на изминаване. Така че, логично е да си мислим, че всяко ребро има тегло, което е двойка (1, *t*) – единица за това, че отиваме в още един град и *t* e времето на изминаване. Съответно можем да пуснем алгоритъма на Дийкстра, който да търси минимален път по тези два приоритета. Ясно е, че той ще намери търсения най-бърз икономичен маршрут. Отново сложността е .

Третата подзадача е за 18 точка. Тя е една от съществените стъпки в решаването на задачата. Не е оценена с малко повече точки, защото има начини, така че решението от втората подзадача да се оптимизира и да хване и тази подзадача. Причината графа да е мулти (може да има повече от една директна магистрална отсечка) между два града е, за да се направят по-лесно тестове, които да забавят предното решение. Първият ни приоритет за маршрутите е да са икономичен – с минимален брой върхове. Това означава, че е логично да търсим тези маршрути с *BFS* от началния град. Въпросът е как да намираме най-бързият от всевъзможните икономични маршрути. Нека дължината на търсения икономичен маршрут е *k*. Ясно е, че предният връх по търсения маршрут от град 1 до град ***N*** ще е на икономичен маршрут от 1 с дължина *k*-1 до предния връх. Освен това трябва и времето на този маршрут да е минимално, за да постигнем минимално време при връх ***N*** след това. Това е и едно от основните свойства на минималния маршрут – всеки негов подмаршрут също е минимален. От казаното следва, че е достатъчно да направим малка модификация на *BFS*. Като се движим от връх на връх, трябва да запазваме и минималното време за достигане до съответния връх, като разбира се движим по икономични маршрути (т.е. се интересуваме само от ребрата, които са между върхове на съседни нива). Може да се забележи, че предната Дийкстра работи почти като сегашния *BFS*, това е и една от причините да е трудно да се покаже разликата по време между двата алгоритъма. Сложността е .

Четвъртата подзадача е за 15 точки. Вече имаме и заявки – предположенията за двойка градове *x* и *y*, за които търсим дали има най-бърз икономичен маршрут от 1 до *y*, минаващ през *х*. Тук имаме улеснение на заявките – знаем, че за всяко предположение град *y* е фиксиран – град ***N***. Сега е един от моментите да използваме нещо от условието – има точно един най-бърз маршрут от 1 до всеки друг град, включително и ***N***. Така че е достатъчно да намерим този маршрут, за да знаем през кои градове минава, т.е. за кои *x* ще имаме отговор и за кои не. Сега остава друг момент – да намерим най-бързият икономичен маршрут между градовете *x* и *y*. Ясно е, че маршрута от *x* до *y*, който е по пътя от 1 до *y* ще е икономичен. Дали ще е най-бързият обаче. От наблюденията във трета подзадача е ясно – трябва да е минимален, иначе нямаше да сме намерили оптималния маршрут от 1 до *y*. Така предпроцесването преди заявките може да е следното – намираме пътя от ***N*** до 1, като за тази цел можем например да си пазим в предишния *BFS* за всеки връх от кой предходен връх се получава най-бързия икономичен маршрут и да се връщаме назад. Като правим това връщане – ще смятаме дължината на пътя до ***N*** и ще сумираме времената на ребрата, за да изчислим и дължината на най-краткия икономичен маршрут. След като сме направили това предпроцесване, можем да отговаряме заявките константно. Така сложността тук ще е .

Петата и последната подзадача е за 36 точки. Вече нямаме улеснения на условието. От трета подзадача стана ясно, че като правим *BFS* ни интересуват само ребрата, които са между върхове от съседни нива спрямо *BFS*-а от началния връх, защото другите ни водят до неикономични маршрути. По принцип този подграф (с тези ребра) на началния е *DAG* – ацикличен ориентиран граф. В него би било трудно да поддържаме бързо описаните заявки, защото имаме множество икономични маршрути от 1 до останалите върхове. Като вземем предвид теглата на ребрата – времената на изминаване, става ясно, че ни интересуват само тези ребра, които ще ни дадат най-бързите икономични маршрути. Обаче в общия случай и това може да е трудно за заявките – може да имаме няколко най-бързи икономични маршрути. Затова е важно наложеното ограничение в задачата – имаме точно един най-бърз икономичен маршрут от 1 до всеки друг град. Това означава, че за всеки връх ще има само едно ребро, което го свързва с връх от предходното ниво. Така от *DAG*, графът може да остане само с тези ребра и да получим дърво, което ще наричаме *BFS* дърво! Както знаем, дърветата имат много хубави свойства – така например е сравнително лесно да отговаряме на заявки дали един връх е предшественик на друг в дървото. Достатъчно да направим един *DFS* от корена, да номерираме върховете по ред на влизане и да запомним за всеки връх времето му на влизане и на излизане. Тогава един връх *x* ще е предшественик на *y*, ако времето на влизане в *y* e в интервала на влизане и излизане от връх *x*. По този начин изяснихме как бързо можем да намираме дали един връх *x* ще се намира на най-бързия икономичен маршрут от 1 до друг връх *y* – търсим дали *x* e предшественик в *BFS* дървото, построено от връх 1 и използвайки времената на ребрата. Остана да изясним как намираме най-краткия икономичен маршрут от *x* до *y*, когато *x* е предшественик на *y* в това дърво. Разсъжденията са подобни на тези в предната подзадача. Ясно е, че пътят между тези два върха в дървото ще е икономичен. Затова трябва той да е и най-бързият, щом дървото е построено правилно и пътят от 1 до *y* e най-бързият икономичен. Крайният алгоритъм се състои накратко от следното – правим *BFS* дърво чрез *BFS* от началния връх, след това с подходяща запазена информация строим дървото и пускаме *DFS* в него за определяне на времената на влизане и излизане и вече сме готови да отговаряме константно на заявките. Окончателната сложност за задачата е .

*Автор: Илиян Йорданов*