Подзадача 1 $N \leq 5; T \leq 5$ **– 15 т.**

За тази подзадача се очаква да мине всяко наивно решение например такова със сложност $O(2^T TN)$. Пробваме всички комбинации полети (2^T) и за всяка наивно проверяваме дали редицата от полети е валидна.

Подзадача 2 $N \leq 2 \times 10^4; T \leq 5 \times 10^2$ **– 15 т. (за общо 30 т.)**

За тази подзадача трябва да се сетим да използваме някакво dp. Най-лесното такова би било по дни/полети. Всъщност имаме два варианта. Можем за всеки предишен полет да пазим най-добри отговор завършващ с този полет и когато гледаме следващия полет да обходим всички предишни и да проверим има ли свързващ ги автобус (линейно). Също така можем да направим обратното, когато разглеждаме нов полет вече да имаме максималния отговор, който позволява да преминем към този полет, после прибавяме едно към него (за самия полет) и накрая обхождаме всички следващи полети и ъпдейтваме отговора им ако има свързващ ги автобус (отново с линейна проверка). Двете решения са приблизително еднакво бързи със сложност $O(T^2 N)$. Всъщност за почти всички подзадачи ще има такива два типа решения. В прикачените решения имената на всички от втория тип ще завършат с `_inv`.

Подзадача 3 $N \leq 1 \times 10^5; T \leq 2.5 \times 10^3$ **– 10 т. (за общо 40 т.)**

Тук използваме доста подобно решения. Целта е да подобрим проверката за свързващ автобус. Има два варианта. Първият е да си сортираме списъка от ребра за всеки от върховете и когато проверяваме за свързаност да използваме двоично търсене. Вторият е да пазим ребрата на всеки от върховете в set и при проверка просто да проверяваме дали се съдържа в set-а. Първото всъщност е доста по-бързо като константа, но и двете са със сложност $O(N \log N + T^2 \log N)$ и вземат тази подзадача.

Подзадача 4 $N \leq 5 \times 10^5; T \leq 5 \times 10^3$ **– 10 т. (за общо 50 т.)**

Тук решението е същото като второто за предната подзадача, но просто използваме unordered_set за да постигнем $O(1)$ проверка и вкарване в set-а. Така сложността става $O(N + T^2)$, но с голяма константа от unordered_set-а.

Подзадача 5 $N \leq 2 \times 10^3; T \leq 5 \times 10^5$ **– 10 т. (за общо 50 т.)**

За тази подзадача трябва да избегнем T^2 сложност в решението си, както се вижда от ограниченията. Можем вместо това да правим dp по върховете като имплицитно то е двумерно dp (с измерения ден/полет и град), но първото измерение не съществува в кода, просто поддържаме dp-то винаги да е в състояние от последния разгледан ден (защото само това ни трябва). Така за всеки град пазим максималния отговор за път завършващ в града или при обратния вариант – максималния отговор за път завършващ в съседен (или същия) град. Това решение има сложност $O(NT)$ и затова хваща и подзадача 3 и назад.

Подзадача 6 $N \leq 5 \times 10^5; T \leq 2 \times 10^4$**– 10 т. (за общо 70 т.)**

Тук целта е да имаме отново сложност както в подзадача 4, но да избегнем лошата константа. Всъщност обаче най-лесният вариант е да използваме решението за предната подзадача. Лесно се вижда, че ако пуснем решението директно то няма да премине в time limit-а заради големия брой върхове. Когато се замислим обаче, повечето върхове (и ребра) не са ни полезни защото не участват в нито един полет (тъй като полетите са доста по-малко от върховете). Т.е. можем да филтрираме ребрата така че да запазим само полезните такива. Сложността става $O(N + \min(N, T)\, T)$ с добра константа.

Подзадача 7 $N \leq 5 \times 10^5; T \leq 5 \times 10^5$**– 30 т. (за общо 100 т.)**

Тази подзадача покрива цялата задача. До сега беше споменато, че за всеки вид решение има два подхода. Нека се концентрираме на решението на подзадача 5. В единия вариант имаме константен ъпдейт за връх, но линейна заявка по съседите му, а в другия обратното – линеен ъпдейт по съседите на връх, но константа заявка за него. Както в много ситуации от подобен характер възможно да се постигне нещо средно, така че комбинацията да е по-бърза в най-лошия случай. Тук това средно е $\sqrt{M}$, но M е от порядъка на N, т.е. решението е със сложност $O\left(N + \sqrt{N}T\right)$. Идеята е да разделим върховете на тежки ($> \sqrt{M}$ ребра) и леки ($< \sqrt{M}$ ребра). При леките можем да си позволим да правим линейна заявка по всичките им съседи със сложност $O\left(\sqrt{M}\right)$. Това обаче не можем да го правим за тежките върхове, т.е. за тях са нужни константни заявки. Постигаме това като всеки връх при ъпдейт ъпдейтва всички свой тежки съседи, забележете, че няма как да има над $M/\sqrt{M} = \sqrt{M}$ тежки върха в целия граф, т.е. и този ъпдейт ще е със същата $O\left(\sqrt{M}\right)$ сложност. Една грешка, която можем да допуснем е да държим всичко в едно и също dp, но това е грешно, защото някои от стойностите представляват максимален отговор от полет кацащ в града (т.е. можем след това да хванем автобус), а други представляват максимален отговор сред съседите на града (т.е. не можем да хванем автобус). Затова поддържаме две dp-та, като второто се използва само за тежките върхове.