Анализ на задача `wfs`

Тагове: динамично програмиране, оптимизации

Решение на първа подзадача - 20 точки

Макар задачата да изглежда на пръв поглед като някакъв вид алчен алгоритъм, то има приятно динамично, което може да моделира ситуацията. Трудното е да установим кой е стейтът, описващ важната информация. Когато правим *BFS* разглеждаме графа на нива. Затова можем да направим следното динамично - $dp[st]$ е минималният брой ребра, които трябва да се добавят, за да получим валидна *BFS* наредба като текущото ниво започва от връх с номер st. Първо, трябва да видим кои върхове задължително трябва да са част от нивото - това са върховете, които са свързани с ребра с някои от $1, 2, \dots, st-1$. Нека максималният номер на такъв връх е m. Тогава в текущото ниво задължително трябва да са върховете с номера $st, st+1, \dots, m$. Ако $m < st$, то можем да считаме, че $m = st$.

Интересен е въпросът какво става с върхове от st нататък, които са съседни на някой по-ранен връх и би трябвало да са в някое от предните нива и няма как да са в текущото. Може да се забележи, че всъщност такива върхове няма как да има, защото те би трябвало да сме ги отчели на по-ранен етап в динамичното (с m).

Сред върховете $st, st+1, \dots, m$ може да има такива, които не са свързани с ребро към някои от предишните върхове. Трябва да пресметнем колко са точно, защото тях задължително трябва да свържем с по едно ново ребро към някои от предишните върхове. Ако означим този брой с c, то получаваме следната рекурентна зависимост:

$$dp[st] = c + \max_{end=m}^{N}(end - m + dp[end + 1])$$

Началната стойност на динамичното е $dp[N + 1] = 0$, а отговорът на задачата е в $dp[2]$ (нулевото ниво се състои само от връх 1, а първото ниво със сигурност започва от връх 2). В тази подзадача можем да реализираме директно това динамично.

Сложност: $O(N(N + M))$.

Решение на втора подзадача - 60 точки

Тук е достатъчно да направим малко преизчисления, за да смятаме по-бързо динамичното. Можем предварително да сметнем m за всеки възможен st - просто изчисляваме максималните номера на всички ребра, излизащи от върховете $1, 2, \dots, st$ последователно за $st = 1, 2, \dots, N-1$. За да смятаме c по-бързо е достатъчно предварително да сметнем за всеки връх минималния номер на съседен връх (така сравнявайки този номер с st разбираме дали ще трябва допълнително ребро). По този начин единствената бавна част остава вътрешния цикъл.

Сложност: $O(M + N^2)$.

Решение на трета подзадача - 100 точки

Можем да разпишем зависимостта по следния начин: $dp[st] = c + \max_{end=m}^{N}(end - m + dp[end + 1]) = c - (m + 1) + \max_{end=m}^{N}(end + 1 + dp[end + 1])$. Затова можем да смятаме $dp'[st] = dp[st] + st$ и $suf[i] = \max_{j=i}^{N} dp'[j]$ и по този начин $dp[st] = c - (m+1) + suf[m+1]$. Можем да считаме, че смятаме динамичните отзад-напред. Вече изяснихме, че предварително сме сметнали m за всяко st, така че единственото трябва да поддържаме бързо c - броят върхове от st до m, които не са съседни на върховете $1, 2, \dots, st-1$. Един прост начин е да имаме множество със свързаните върхове, защото те малко по-малко спират да бъдат свързани (с намаляване на st). На всяка итерация добавяме нов елемент в множеството, потенциално махаме

някои от множеството заради намаляването на m и потенциално трябва да видим кои спират да бъдат свързани. Друг подобен начин, но по-бърз е да поддържаме върховете от множеството с булев масив и предварително да видим на дадена позиция кои от върховете спират да бъдат свързани.

Сложност: $O(M + N\log_2 N)$ или $O(N + M)$.

Автор: Илиян Йорданов (заета)