# АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ПЪТИЩА

**Подзадача 1.**
Пълно изчерпване на всички пътища ще доведе до работещо решение. Можем да тръгнем от връх X, и да търсим пътища с дължина L до Y. За да сметнем тази стойност е достатъчно да търсим рекурсивно броят на пътищата с дължина L-1, тръгващи от съсед на X, които стигат до Y. Внимателно изчисляване на броят на стъпките, като се вземат под внимание особеностите на графа от условието, показват че ограниченията N <= 10, M <= 15, L <= 12, позволяват на това решение да работи.

**Подзадача 2.**
Предното решение изискваше рекурсивно изчисление на стойностите на функция *solve(vertex, length)* , която ни дава броя на пътищата от връх vertex с дължина length до върха Y. В подзадача 2 можем да използваме динамично оптимиране (или мемоизация), за да избегнем изчисляването на solve(vertex, length) повече от веднъж. Сложността на решението е O(L * M). За фиксирана дължина, трябва да обходим всички изходящи ребра по веднъж.

**Подзадача 3.**
Опитните състезатели би трябвало бързо да видят „стандартното“ решение на задачата. Да разгледаме матрицата на съседство A на дадения граф. Стойността на клетката A[i][j] ще ни даде броя на пътищата от връх i до връх j с дължина едно (това е или 0, или 1). Да разгледаме броя на пътищата с дължина 2 от връх i до връх j. Този брой може да се изрази като A[i][1] * A[1][j] + A[i][2] * A[2][j] + … + A[i][N] * A[N][j]. (Ще имаме път с дължина две през връх k само ако имаме ребра (i, k) и (k, j). Това се случва точно когато A[i][k] = A[k][j] = 1). Това е точно формулата за елемента B[i][j] на матрицата B = $A^2$. Именно, $A^2$[i][j] ще ни даде броя на пътищата с дължина 2 между върховете i и j.
Подобно, броят на пътищата с дължина 3 между i и j ще е B[i][1] * A[1][j] + … + B[i][N] * A[N][j]. Тоест, точно $A^3$[i][j].
Отговорът на задачата в тази подзадача ще е $A^L$[X][Y]. Изчисляването на $A^L$ е възможно с алгоритъм за бързо повдигане на степен. (вижте „Програмиране=++Алгоритми;“ от Преслав Наков, достъпна безплатно на http://www.programirane.org/download/ . Глава 7.5) Сложност на решението O($N^3$ * log(L))

**Подзадача 4.**
За да решим Подзадача 4. вече трябва да направим наблюдения, специфични за конкретната задача. Нека означим степента на всеки връх в нашия граф с **k**, броя на общите съседи на два свързани върха с **a**, и броя на общите съседи на два различни несвързани върха с **b**. Да разгледаме отново пътищата с дължина 2 между два върха **i** и **j**. Имаме 3 случая:

1. Ако **i** = **j**, имаме **k** пътя.
2. Ако **i** и **j** са свързани с ребро: тогава имаме **a** пътя (по един път за всеки общ съсед)
3. Ако **i** и **j** не са свързани с ребро: тогава имаме **b** пътя.

Броят на пътищата с дължина 2 не зависи от конкретния избор на **i** и **j**. Както забелязахме по-рано, броят на пътищата с дължина две е еквивалентен на елемент на $A^2$, матрицата на съседство повдигната на втора степен. Стойностите на елементите на $A^2$ ще зависят само от това дали двата елемента са еднакви, и дали са свързани с ребро или не. Тоест ще можем да напишем формула за $A^2$, която зависи от матриците A, I, J. I е матрицата с всички елементи равни на 0, освен елементите на главния диагонал (I[x][x] = 1 ; I[x][y] = 0, ако x != y). J е матрицата, всички елементи на която са равни на 1.
Търсената формула е $A^2$ = (**a** – **b**) * A + **b** * J + (**k** – **b**) * I. Коефициентите могат да се намерят като се реши система уравнения.

Твърдение: $A^L$ може да се представи във вида $\mathbf{x} * A + \mathbf{y} * J + \mathbf{z} * I$, където **x, y, z** са цели числа.
Доказателство по индукция: Вярно е за $A^1$, $A^2$ (виж по-горе).
Да допуснем, че е вярно за $A^K = \mathbf{x} * A + \mathbf{y} * J + \mathbf{z} * I$. Ще покажем вида на $A^{K+1}$ (използвайки прости свойства на умножението на матрици).
$A^{K+1} = \mathbf{x} * A^2 + \mathbf{y} * J * A + \mathbf{z} * I * A$
$A^{K+1} = \mathbf{x} * (\mathbf{a} - \mathbf{b}) * A + \mathbf{x} * \mathbf{b} * J + \mathbf{x} * (\mathbf{k} - \mathbf{b}) * I + \mathbf{y} * \mathbf{k} * J + \mathbf{z} * A$
$A^{K+1} = (\mathbf{x} * (\mathbf{a} - \mathbf{b}) + \mathbf{z}) * A + (\mathbf{x} * \mathbf{b} + \mathbf{y} * \mathbf{k}) * J + \mathbf{x} * (\mathbf{k} - \mathbf{b}) * I$
Следователно, $A^L$ се представя във вида $\mathbf{x} * A + \mathbf{y} * J + \mathbf{z} * I$. Намирането на коефициентите може да се направи чрез изчисляване на елементи на 3 зависещи една от друга линейни рекурентни редици (Вижте http://www.codechef.com/wiki/tutorial-dynamic-programming#Finding_nth_Finobacci_number_in_Olog_n и задача Аминосупа от Зимни състезания, 2009).

А именно рекурентните редици са
$\mathbf{x_i} = \mathbf{x_{i-1}} * (\mathbf{a} - \mathbf{b}) + \mathbf{z_{i-1}}$
$\mathbf{y_i} = \mathbf{x_{i-1}} * \mathbf{b} + \mathbf{y_{i-1}} * \mathbf{k}$
$\mathbf{z_i} = \mathbf{x_{i-1}} * (\mathbf{k} - \mathbf{b})$

По този начин можем да изчислим $A^L$ със сложност $O(3^3 * \log(L) + N^2)$.

**Подзадача 5.**
Достатъчно е да забележим, че в решението на подзадача 4, ние изчислихме цялата матрица $A^L$ като всъщност се интересуваме само от един елемент на матрицата. Достатъчно е да изчислим само елемента на $A^L$, от който се интересуваме, за да решим задачата. Сложността на решението ще бъде $O(3^3 * \log(L))$
Остава да намерим по бърз начин параметрите на графа **k, a, b**, но това не би трябвало да е проблем.

Еквивалентно, можем да забележим, че броят пътища с дължина K между два върха зависи само от това дали върховете съвпадат, или дали са свързани или не с ребро. Можем да дефинираме три линейни рекурентни редици **c, d, e**, определящи броят на пътищата с определена дължина ако 1два върха са съседи, не са съседи, или съвпадат.
Конкретно:
$\mathbf{c_1} = 1$
$\mathbf{d_1} = 0$
$\mathbf{e_1} = 0$
$\mathbf{c_n} = \mathbf{a} * \mathbf{c_{n-1}} + (\mathbf{k} - \mathbf{a} - 1) * \mathbf{d_{n-1}} + \mathbf{e_{n-1}}$
$\mathbf{d_n} = \mathbf{b} * \mathbf{c_{n-1}} + (\mathbf{k} - \mathbf{b}) * \mathbf{d_{n-1}}$
$\mathbf{e_n} = \mathbf{k} * \mathbf{c_{n-1}}$

Допълнение: Графите от задачата са още известни като Силно регулярни графи. За повече информация може да видите, http://en.wikipedia.org/wiki/Strongly_regular_graph

*Автор: Йордан Чапъров*