АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА НАСТАНЯВАНЕ

Задачата се решава чрез използване на два основни алгоритми: BFS и така наречената „задача за раницата“. Може би най-объркващото в задачата е входът, който затруднява директното прилагане на тези алгоритми. Затова ще номерираме имената на участниците с числата от 1 до *N*, използвайки за тази цел структура от данни *map*, която ще наречем *idOf*. Всеки път когато въведем имената на двамата приятели, ние извикваме функцията *get_id()*, която връща индекса на името. Във функцията се извършват две основни неща за всяко име: първо се променят всички главни букви в малки такива, за да се спази условието, според което тази особеност на буквите се игнорира. След това се проверява дали това име вече е срещано преди, като го търсим в *idOf* и ако е срещано, връщаме вече запаметения индекс, иначе го добавяме в *idOf* и връщаме броя на вече използваните индекси до сега плюс 1.

След като вече сме получили индексите на всяко име, построяваме граф *gr*, който ще представя приятелствата. В него пускаме BFS, който намира броя на компонентите му, в случая „кръговете от приятелства“, както и от колко човека се състои всеки от тях. Следва прилагане на „задачата за раницата“. За целта ще са ни нужни два вектора *possible* и *sums*. Клетката *possible[i]* ($0 \leq i \leq N$) ще има стойност:

- 1, ако сумата *i* може да се направи като сбор от големините на един или повече крът от приятелства;
- 0, ако съответната сума е невъзможна.

Тук на помощ идва и sums, в който записваме всички суми, които се направили до сега. В началото на sums добавяме 0 като възможна сума. Попълването на possible става чрез обхождане на големините на кръговете от приятелства, които предварително сме записали във вектора *friendships*. За всеки елемент *i* от *friendships*, преминавайки през всяка от записаните до сега суми в *sums*, образуваме новата възможна сума *newSum* = *sums[j]* + *i* (*sums[j]* обозначава клетката от *sums*, която разглеждаме в момента.) За да не добавяме еднакви суми отново, проверяваме дали *possible[newSum]* е 1, и ако не е:

- Правим *possible[newSum]* = 1;
- Добавяме в *sums* новополучената сума *newSum*;
- При условие, че $newSum \leq V_1$ и $N - newSum \leq V_2$ (тоест можем да настаним newSum участника в първия хотел и N-newSum във втория), намираме най-евтината от образуваните до сега цени и получената сега *answer* = *min(answer, price)*, като $price = newSum * P_1 + (N - newSum) * P_2$.

Също така разглеждаме и тривиалните случаи, където в един от двата хотела не е настанен нито един участник.

Задачата с дадените ограничения работи и с динамичното решение със сложност N^2, в което търсим колко хора могат да се настанят в първия хотел, а останалите настаняваме във втория и така намираме най-евтината цена. Това решение е описано в `best-n^2.cpp`.

Автори: Александър Иванов и Йоана Зелова