Анализ на задача bus

Приготвен от Веселин Георгиев.

1. Авторско решение

Задачата е замислена да бъде решена чрез динамично оптимиране. Преди да изведем целевата функция, която описва решението, ще се спрем на един подпроблем, а именно: как да пресметнем човекоминутите за подинтервал $[i, j]$, ако знаем, че на позиции i и j има спирки, и други спирки в интервала няма?

За целта ще си дефинираме функцията $g(i, j)$, която дава отговор на този въпрос. Очевидно, ако задачата бъде зададена с $K = 2$ (т.е. разрешено е да се сложат само 2 спирки – началната и крайната), то отговора на задачата ще бъде точно $g(1, n)$.

Дефиниция:

$$g(i, j) := \sum_{k=i+1}^{j-1} c_k * min(k-i, j-k)$$

Тъй ще ни трябва да пресмятаме тази функция многократно, е удобно да си въведем един масив $g[][]$, който да държи стойностите пресметнати предварително.

Тук се натъкваме на проблем: пресмятането на всички стойности, по дефиницията, ще отнеме време от порядъка на $O(N^3)$, което е неприемливо (задачата допуска само $O(N^2)$ решение, виж по-долу).

Това може да се избегне, чрез следната рекурсивна дефиниция:

g(i, j) := 0 , при $i <= j + 1$;
g(i, j) := g(i+1, j-1) + sum(i+1, j-1) , иначе.

Където със sum(i, j) сме отбелязали сумиране на елементите c_k за k от i до j, включително. Сега, сумите sum(i, j) могат да се смятат с константна сложност, ако се използва един масив SUM[], като SUM[i] ще пази сумата на всички елементи на c_k за k от 1 до i включително.

sum(i, j) := SUM[j] – SUM[i-1], като додефинираме SUM[0] = 0.

Вижда се, че пресмятането на всеки от елементите на $g(i, j)$ става с константна сложност (едно обръщение до вече пресметната стойност на $g(i, j)$ и обръщение до масива SUM). Т.е. всичките N^2 стойности на $g(i, j)$ могат да се сметнат общо за $O(N^2)$ време.

За решение на оригиналната задача, дефинираме функцията $f(i, l)$, която ще ни казва, колко най-малко времеминути можем да постигнем за подинтервала $[i, n]$, ако последно сме сложили спирка на позиция i и ни остават да разположим още l спирки на позициите между i и n (и все още между i и n няма други спирки).

Очевидно, отговорът на задачата се получава, като пресметнем $f(1, k\text{-}2)$ (вадим 2 от k, за спирките в началото и края на маршрута, които са задължителни по условие).

Функцията f се дефинира така:

f(i, l) := g(i, n) , ако $l = 0$
f(i, l) := 0 , ако $i >= n - 1$
f(i, l) := min [g(i, k) + f(k, l – 1)] за всички k от i + 1 до n – 1 включително , иначе

Тази дефиниция, макар и вярна, няма да ни свърши работа: всяко извикване на функцията *f*(*i*, *l*) поражда извикването на още O(*N*) екземпляра на същата функция. Този проблем се решава по стандартната схема за динамичното оптимиране: чрез *memoization*, осигуряваме всяко изчисление на *f*(*i*, *l*) да се изпълнява само веднъж.

Оценката на сложността на горното решение е $O(K*N^2)$. Наистина, имаме *K***N* различни стойности на функцията, всяка от които се пресмята за O(*N*) време (заради вътрешния цикъл). Ако K е близо до *N*, сложността става около $O(N^3)$. Това решение е неприемливо бавно, тестовете са подбрани така, че решението да хване около 50-60 точки. За да го ускорим, ще приложим техниката "съкращаване на вътрешния цикъл", често използвана при задачи с динамично оптимиране. В конкретния случай, техниката се състои в това, да не въртим вътрешния цикъл по *k* от *i* + 1 до *n* – 1, а да го прекратим по-рано. Наистина, нека разгледаме една реална ситуация, в която хората са равномерно разпределени по всички позиции. Няма логика, след като сме поставили спирка на позиция *i*, да слагаме следващата спирка прекалено далеч от *i* – ако ни остават X позиции за слагане на M спирки, то е логично да слагаме спирките през интервали от X/M позиции.

Как обаче да въведем тази логика в конкретното решение? Нека предположим, че решаваме *f*(*i*, *l*). Пробвали сме, какво ще стане, ако сложим спирка на позиция $k = i + 1$. Това е било съпроводено с извикване на *f*(*i*+1, *l* – 1). Нека предположим, че изпълнението на *f*(*i* + 1, *l* – 1) е приключило, и е открито, че е най-подходящо да сложим следващата спирка на позиция X. Сега, да се върнем на *f*(*i*, *l*). Твърдим, че няма смисъл да поставяме спирката по-нататък от X: наистина, да предположим, че го направим, като я сложим на позиция X + 1. Тогава, оставащите *l* – 1 спирки трябва да "разхвърляме" между позиции X + 2 до *N* – 1 включително. Да, но вече знаем, че няма да получим по-добро решение за *f*(*i* + 1, *l* – 1), ако сложим следващата спирка на позиция след X. Това е противоречие с допускането, че има смисъл да сложим спирката на позиция след X, при извикването на *f*(*i*, *l*).
След прилагането на тази оптимизация, решението върви доста по-бързо. Например, ако приемем, че оптималното решение на задачата е да разпределим спирките равномерно по маршрута на автобуса, то за решаването на всяка от *N* * *K* стойности на функцията са необходими около *N*/*K* итерации на вътрешния цикъл, т.е. сложността тук става $O(N^2)$. Ако, обаче, хората в позициите са силно неравномерно разположени, оптимизацията помага малко и решението отива към $O(N^3)$. Тъй като тази сложност няма как да бъде избегната, а и в реалния живот рядко ще имаме подобен сценарии, просто няма такива тестове (силно неравномерни). Надявам се състезателите да не се излъжат много да търсят "истинско" N^2 решение.

2. Други решения

- **Изчерпване**: очаква се да вземе 10-20 точки
- **Greedy** (лаком алгоритъм): Следното просто greedy изглежда да "върши работа". Слагаме по една спирка в началото и края, останалите позиции (без крайните) се сортират по брой жители в обратен ред. След което на първите *k*-2 позиции (тези с най-много жители) се слагат спирки. Макар и очевидно невярно, това решение се пише много лесно и се справя учудващо добре на реални тестове. В приложените от автора, обаче, това решение хваща точно 30 точки.
- **Динамично**, $O(N^3)$: Очаква се да хване от 40-60, в зависимост от реализацията.