СЛОЖНОСТ НА АЛГОРИТМИЧНИ ЗАДАЧИ

КОНТРОЛНО № 3 ПО "ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ" — СУ, ФМИ, 09. 01. 2019 Г.

Задача 1. Формулираме алгоритмичната задача `PointsInPlane` така: n точки в равнината, зададени с координатите си, да се наредят по нарастващ ред на разстоянието си до т. $O(0;0)$. Докажете, че ако се решава чрез сравнения, задачата има времева сложност $\Theta(n \log n)$. За целта:
а) съставете алгоритъм със сложност $O(n \log n)$; **(5 точки)**
б) докажете долната граница $\Omega(n \log n)$ чрез подходяща редукция. **(15 точки)**

Задача 2. Нека $f(n)$ е времевата сложност на задачата за умножение на две n-цифрени числа, а $g(n)$ е времевата сложност на задачата за намиране на квадрата на n-цифрено число. Тези две сложности са неизвестни поотделно. Докажете, че $g(n) = \Theta(f(n))$. За целта:
а) докажете, че $f(n) = \Omega(n)$ и $g(n) = \Omega(n)$; **(5 точки)**
б) докажете, че $g(n) = O(f(n))$; **(5 точки)**
в) докажете, че $g(n) = \Omega(f(n))$. **(10 точки)**

РЕШЕНИЯ

Задача 1. а) Алгоритъм със сложност $O(n \log n)$: сортираме точките по разстояние до т. $O(0;0)$ с помощта на някоя от бързите сортировки, например чрез пирамидално сортиране. Разстоянията се пресмятат по известната формула от геометрията. Пресмятането на всяко отделно разстояние става за константно време; на всички разстояния общо — за време $\Theta(n)$. Същинското сортиране се извършва за време $\Theta(n \log n)$. Окончателно, времето на целия алгоритъм е $\Theta(n \log n)$. Ето защо задачата `PointsInPlane` може да се реши за време $O(n \log n)$.

б) Долната граница $\Omega(n \log n)$ на задачата `PointsInPlane` се доказва с помощта на редукция от задачата `SortPositive` за сортиране на масив A[1...n] от положителни числа, за която знаем, че изисква време $\Omega(n \log n)$, когато се решава чрез сравнения.

```
SortPositive(A[1...n])
P[1...n]: array of points in the plane
for k ← 1 to n do
   P[k].x ← A[k]
   P[k].y ← 0
PointsInPlane(P[1...n])
for k ← 1 to n do
   A[k] ← P[k].x
```

Коректност на редукцията: Точката P[k] изобразява числото A[k] върху положителната част на абсцисната ос, затова разстоянието от т. P[k] до т. O е равно на A[k]. Функцията `PointsInPlane` подрежда точките по разстоянията им от нулата, тоест подрежда числата A[k] в нарастващ ред.

Бързина на редукцията: Двата цикъла имат обща времева сложност $\Theta(n) = o(n \log n)$, затова редукцията е достатъчно бърза за целите на доказателството.

Задача 2. а) Това са тривиални долни граници по размера на входа: всеки алгоритъм трябва поне да прочете цифрите на числата.

б) Повдигането на квадрат е частен случай на умножение. Частният случай е не по-труден от общия, затова $g(n) = O(f(n))$.

в) Обратно, $g(n) = \Omega(f(n))$, тоест $f(n) = \mathrm{O}(g(n))$, защото умножението може да се сведе до повдигане на квадрат по формулата $xy = \frac{(x+y)^2 - (x-y)^2}{4}$. Останалите аритметични операции в тази формула — делението на 4, събирането и изваждането — изискват време $\Theta(n)$. Ето защо тази формула представлява алгоритъм за умножение на две n-цифрени числа с времева сложност $\Theta(g(n)) + \Theta(n) = \Theta(g(n))$, защото $g(n) = \Omega(n)$ според доказаното в подусловие "а". А щом има алгоритъм за умножение със сложност $\Theta(g(n))$, тогава времевата сложност на умножението е $f(n) = \mathrm{O}(g(n))$.

ТОЧКУВАНЕ

Задача 1.

а) 5 точки за описание на алгоритъм с времева сложност $\mathrm{O}(n \log n)$;

б) 15 точки за доказване на долната граница, от които по 5 точки за всяка от следните три стъпки:
- — описание на редукцията;
- — доказателство за коректност на редукцията;
- — анализ на бързината на редукцията.

Задача 2.

а) 5 точки за позоваване на тривиалните долни граници по размера на входа;

б) 5 точки за позоваване на връзката между общия и частния случай;

в) 10 точки, от които по 5 точки за всяка от следните две стъпки:
- — свеждане на умножението до повдигане на квадрат;
- — анализ на бързината на алгоритъма.