Олімпіада з computer science

2 травня 2017 р.

Розв’язки задач

1. Кола

Завдання 1. Назвімо кожну точку *лівою* або *правою* залежно від того, є вона лівою чи правою межею кола. Розглянемо найлівішу праву точку. Коло, правою межею якого вона є, за ліву межу може мати лише сусідню з нею зліва точку. Приберемо ці дві точки і коло, що через них проходить, із набору і повторимо ту саму операцію ще раз, а потім повторюватимемо далі, доки залишаються точки. Таким чином нам вдасться однозначно визначити увесь набір кіл.

Завдання 2. Один з можливих підходів — побудувати дерево, у якому вершини — це кола і вершина A є дочірньою іншої вершини B, якщо коло, що відповідає вершині A, вкладене у коло, що відповідає вершині B. При цьому вершина-корінь — це фіктивне коло, що умовно намальовано навколо всіх інших кіл набору. Тоді ступінь відрізаності одне від одного двох кіл — це відстань між відповідними вершинами у дереві, зменшена на 1, якщо одне з кіл є дочірнім іншого, та зменшена на 2 в інших випадках.

Обійдемо дерево пошуком у глибину та, починаючи з найнижчих вершин (листя), зберігатимемо для кожної вершини глибину найглибшого її піддерева (разом із найглибшою вершиною), а також глибину другого за глибиною піддерева (з відповідною найглибшою вершиною). За цією інформацією неважко відновити довжину найдовшого шляху між парою вершин з даного піддерева за умови, що він проходить через корінь; так само можна відновити і самі дві вершини, шлях між якими найдовший, тобто пару кіл найбільшого ступеня відрізаності одне від одного. Інформація про глибину найглибшого піддерева, зібрана для дочірніх вершин, дозволяє відразу відновити потрібну інформацію і для батьківської, тому увесь обхід займе лінійний від кількості вершин час. Відповідь — одна з розглянутих під час обходу пар — та, ступінь відрізаності одне від одного кіл у якій найбільша.

2. Сортувальний автомат

Завдання 1. Програма може бути такою:

- Якщо у пам’яті автомата записано нуль:
  - якщо це клітинка N, завершити роботу;
  - інакше виконати три операції: піти праворуч, обміняти місцями числа і піти ліворуч.
- Інакше, якщо у пам’яті автомата записане число, менше за те, що стоїть у клітинці:
  - якщо це перша клітинка, то виконати одну операцію: обміняти числа;
  - інакше виконати одну операцію: піти ліворуч.
- Інакше виконати дві операції: піти праворуч та обміняти місцями числа.

Даний алгоритм працює «ітераціями»: спочатку розташовує у порядку неспадання перші два числа, потім перші три, далі перші чотири і т. д. По завершенні ітерації, на якій відсортовано перші K чисел, автомат стає у клітинку з номером K, а в його пам’яті записано число 0. Після цього автомат переходить до ітерації $K + 1$: бере вміст наступної клітинки та переміщує його ліворуч доти, доки числа у клітинках, по яких проходить автомат, більші за дане. Щойно це правило порушується, автомат ставить число на відповідне місце і за принципом доміно переставляє числа на одне місце праворуч, доходячи таким чином до кінцевої позиції $K + 1$.

Завдання 2. Програма, наведена вище, на кожній ітерації K виконує лінійну від K кількість операцій. Загалом, таким чином, буде виконано квадратичну від N кількість операцій. Щоб довести, що ефективнішої програми, яка розв’язує поставлену задачу, не існує, розглянемо для довільного N початкову послідовність чисел N, $N – 1$, $N – 2$, ..., 3, 2, 1. Автомат має так чи інакше витратити $2(N – 1)$ операцій на те, щоб переставити число N на останнє місце, а одиницю — на перше; $2(N – 3)$ операцій на те, щоб переставити на передостаннє місце число $N – 1$, а на друге — число 2; $2(N – 5)$ операцій на те, щоб поміняти місцями числа $N – 2$ та 3, і т. д. Тут сумарна кількість операцій також є квадратичною функцією від N.

3. Найбільший спільний дільник

Завдання 1. Для довільних натуральних $x < y$ справджується нерівність $x + (y \bmod x) \leqslant y$ (де через $y \bmod x$ позначено остачу від ділення y на x). Тому за два кроки пару чисел (A, B), де $A < B$, алгоритм Евкліда перетворить на пару із сумою, що не перевищує A. Таким чином, через кожні два кроки сума чисел у парі зменшується принаймні у два рази. Тоді загальна кількість кроків, які виконає алгоритм, не перевищує подвоєного логарифма суми початкової пари.

Тепер розглянемо послідовність Фібоначчі — числа 1, 2, 3, 5, 8, 13, ..., кожне наступне з яких дорівнює сумі двох попередніх. Неважко зрозуміти, що якщо на вхід алгоритму передати N-й та $(N + 1)$-й члени цієї послідовності, то алгоритм переходитиме щоразу до попередньої пари сусідніх чисел Фібоначчі і прийде до пари (1, 0) якраз після N кроків. З іншого боку, N-й член послідовності Фібоначчі F_N менший за 2^N (довести це можна за індукцією: якщо $F_{N-1} < 2^{N-1}$ та $F_N < 2^N$, то $F_{N+1} = F_N + F_{N-1} < 2^N + 2^{N-1} < 2^{N+1}$). Тому сума N-го та $(N + 1)$-го членів менша за 2^{N+2}, а кількість кроків, які виконає алгоритм, — не менша ніж логарифм суми, зменшений на 2. Від'ємник-двійка, звичайно, не є принциповим, адже для достатньо великих N його можна компенсувати незначним зменшенням числа c (див. примітку до умови задачі).

Завдання 2. Задамо алгоритм таким чином:

- Якщо одне з чисел вхідної пари (A, B) нульове, повертаємо інше число.
- Якщо A парне, а B непарне, перетворюємо пару (A, B) на $(A/2, B)$ — НСД цієї пари збігається із шуканим. Аналогічно діємо у випадку, коли A непарне, а B парне.
- Якщо обидва числа є парними, перетворюємо пару (A, B) на $(A/2, B/2)$, а кінцевий результат домножуємо на 2.
- Якщо обидва числа є непарними і $A \leqslant B$, перетворюємо пару (A, B) на $(A, (B - A)/2)$.

Щоб показати, що час виконання даного алгоритму є логарифмічним від суми вхідних чисел A і B $(A \leqslant B)$, зауважимо, що після кожного кроку добуток чисел пари зменшується принаймні удвічі. Тоді загальна кількість кроків не перевищує

$$\log_2(AB) = \log_2(A) + \log_2(B) \leqslant 2\log_2(B) \leqslant 2\log_2(A + B).$$