Homework #5

Замітка. Через $\mathbb{Z}_+$ буду позначати $\mathbb{N} \cup \{0\}$.

Завдання 1.

Пункт А. Згідно матеріалу лекції, кількість операцій Q, яка потрібна для знаходження F_n згідно алгоритму 1 дорівнює F_n, яке може бути знайдене згідно формули

$$Q = \frac{1}{\sqrt{5}}\left(\varphi^n + \psi^n\right)$$

Де $\varphi = \frac{1+\sqrt{5}}{2}, \psi = \frac{1-\sqrt{5}}{2}$.

Додатком ψ^n можна знехтувати для великих n, тому маємо порядка $Q = \frac{1}{\sqrt{5}} \cdot \varphi^n$ операцій.

Оцінимо n, для яких буде ще "раціональний" час виконання програми. Зазвичай, емпіративно, вже деякі "затримки"/проблеми починаються десь приблизно з 1 млрд операцій, тобто $Q_{\text{crit}} = 10^9$. Тому відповідне n_{crit} має значення

$$n_{\text{crit}} = \frac{\ln(\sqrt{5}Q_{\text{crit}})}{\ln \varphi} \approx 45$$

Це число дійсно вже дуже погано обчислюється, якщо алгоритмом 1 знаходити F_{45}.

Пункт Б. Помітимо, що алгоритм 2 виконує порядка n дій. Також позначимо швидкість виконання $\nu = 10^3\ s^{-1}$.

Для першого алгоритма маємо близько $Q_1 = \frac{1}{\sqrt{5}}\varphi^n$ операцій (для оцінки можна було і додатком $1/\sqrt{5}$ знехтувати, але нехай залишається). Отже час, який нам потрібен:

$$\tau_1 = \frac{Q_1}{\nu} = \frac{\varphi^n}{\sqrt{5}\nu} \approx \frac{1.6^{1000000}}{1000\sqrt{5}}$$

Звісно це число дуже велике і стандартний калькулятор навіть не може це порахувати 🙂

Другий алгоритм ж має час виконання:

$$\tau_2 = \frac{Q_2}{\nu} = \frac{n}{\nu} = \frac{10^6}{10^3\ s^{-1}} = 10^3\ s$$

Це приблизно 16.6 хвилин. Звісно, 1000 операцій в секунду — це дуже повільно по сучасним стандартам, але це в рази краще алгоритма 1, у якого час навіть не знаходиться на калькуляторі.

Якщо збільшити ν, то звісно це не сильно вплине на відносне порівняння τ_1 та τ_2, оскільки час виконання алгоритму 1 все ще буде величезним. Ну а алгоритм 2 буде виконуватись 1.6 хвилин та 0.6 хвилин для нових значень ν, відповідно, що звісно приємне покращення.

Отже легко бачити висновок: алгоритм 2 зі складністю $\mathcal{O}(n)$ в рази швидше працює, ніж алгоритм 1 зі складністю $\mathcal{O}(\varphi^n)$.

Завдання 2.

Отже, запропонуємо такий алгоритм для $a, b \in \mathbb{Z}_+$ (можна і узагальнити для від'ємних чисел, просто вважаючі $\gcd(a, b) = \gcd(|a|, |b|)$). Нехай ми запустили функцію $\gcd(a, b)$ і на деякому етапі маємо пару (i, j):

1. Якщо i та j ділиться навпіл, то повертаємо $\gcd(\frac{i}{2}, \frac{j}{2})$.
2. Якщо i ділиться навпіл, а j — ні, то повертаємо $\gcd(\frac{i}{2}, j)$, аналогічно для випадку, коли i не ділиться, а j ділиться.
3. Якщо обидва числа непарні, то повертаємо $\gcd(\frac{|i-j|}{2}, \min\{i, j\})$. Це твердження неочевидне, тому покажемо, що воно дійсно виконується.

   Твердження. Якщо i, j обидва непарні, то $\gcd(i, j) = \gcd(\frac{|i-j|}{2}, \min\{i, j\})$

   Доведення. Спочатку скористаємось тим, що $\gcd(i, j) = \gcd(\max\{i, j\}, \min\{i, j\})$, тобто поміняємо a, b місцями, якщо $a < b$. Це твердження виконується, оскільки знаходження НСД є комутативною операцією.

   Далі $\gcd(\max\{i, j\}, \min\{i, j\}) = \gcd(\max\{i, j\} - \min\{i, j\}, \min\{i, j\})$. Тут ми користаємось властивістю $\gcd(a, b) =$

$\gcd(a-b,b)$, яку доводили раніше. Оскільки $\max\{i,j\}-\min\{i,j\} = |i-j|$, то $\gcd(i,j) = \gcd(|i-j|, \min\{i,j\})$.

Нарешті помічаємо, що $|i-j|$ парне, оскільки i, j обидва непарні. А отже, користуючись властивостями з завдання, маємо $\gcd(i,j) = \gcd(\frac{|i-j|}{2}, \min\{i,j\})$.

На мові *Python* реалізація має вид

```
def gcd(a,b):
    if a == 0 or b == 0:
        return max(a,b)
    if a % 2 == 0 and b % 2 == 0:
        return 2*gcd(a//2,b//2)
    if a % 2 == 0 or b % 2 == 0:
        return gcd(a//2,b)
    if a % 2 == 1 and b % 2 == 0:
        return gcd(a,b//2)
    return gcd(abs(a-b)//2, min(a,b))
```

Майже впевнений, що реалізацію можна якось скоротити, але так вона виглядає ідейно простою. Помітимо, що тут реалізація рекурсивна, проте її можна реалізувати і без використання рекурсії.

Доведення коректності та часткової коректності.

Доведемо часткову коректність. Інваріантом рекурсії (або циклу у нерекурсивній реалізації) є *НСД* двох чисел a, b. На кожному кроці він зберігається і це доведено зверху. Тому якщо остаточно ми отримаємо $i = 0$ або $j = 0$, то $\max\{i,j\}$ дійсно буде дорівнювати $\gcd(a,b)$.

Доведемо повну коректність, тобто що алгоритм завершить свою роботу. Для цього пропоную довести наступне: після кожної операції сума $i + j$ строго зменшується в разі, якщо i, j обидва не нулі. В такому разі алгоритм повністю коректний. Це випливає з твердження знизу.

Твердження. Нехай маємо функцію $f : \mathbb{Z}_+^2 \to \mathbb{Z}^2$. Тоді якщо $\forall i, j \in \mathbb{N}$, $f(i,j) = (i', j')$ виконується $i' + j' < i + j$, то $\exists k \in \mathbb{N}, f^k(i,j) = (v,u)$, причому $v = 0 \vee u = 0$, де позначка f^k позначає $f^k := \underbrace{f \circ f \circ f \circ \cdots \circ f}_{k \text{ times}}$.

Доведення. Нехай $f^k(i,j) = (i_k, j_k)$. Позначимо $\omega_k := i_k + j_k$. Тоді згідно означенню послідовність цих сум Ω є строго спадаючою, причому $\omega_0 = i_0 + j_0 \in \mathbb{N}$.

Тепер будемо доводити від супротивного. Нехай

$$\forall k \in \mathbb{N}, f^k(i,j) = (i_k, j_k),\ i_k \neq 0 \land j_k \neq 0$$

Враховуючі умову зверху, мінімальне значення, що може приймати Ω — це 2, що відповідає парі $(i_k, j_k) = (1,1)$. Проте $f(i_k, j_k) = (i_{k+1}, j_{k+1})$, де $i_{k+1} + j_{k+1} < 2$, тому $\omega_{k+1} < 2$, тому $\min \Omega = \omega_{k+1} < 2$, що суперечить тому, що $\min \Omega = 2$.

Повернімося до доведення повної коректості. Доведемо, що після кожної операції $i + j$ строго зменшується.

Якщо хоча б одне з чисел парне, то ми вочевидь строго понижуємо одне або обидва значення з пари (a, b), оскільки ми ділимо одне або обидва значення на 2, тому і суму відповідно ми теж строго понижуємо.

Доведемо, що те саме ми отримуємо, коли робимо операцію з двома непарними числами. Нехай $i = 2n+1, j = 2k+1$. Тоді з пари $(2n+1, 2k+1)$ ми отримуємо пару чисел $(|n-k|, 2\min\{n,k\}+1)$. Сума до операції і після операції, відповідно:

$$\omega_0 = 2(n+k+1),\ \omega = |n-k| + 2\min\{n,k\} + 1$$

Розглянемо різницю:

$$\Delta\omega = \omega_0 - \omega = 2n + 2k + |n-k| - 2\min\{n,k\} + 1$$

Помітимо, що $n + k - \min\{n,k\} = \max\{n,k\}$, тому

$$\Delta\omega = 2\max\{n,k\} + |n-k| + 1 > 0$$

Тому бачимо, що $\omega_0 > \omega\ \forall i,j \in \mathbb{Z}_+$. Повну коректність доведено.

Складність алгоритму

Розглянемо послідовність сум $\omega_k := i_k + j_k$, де (i_k, j_k) — пара чисел після k виконань алгоритму. Позначимо $\varepsilon_k := \frac{\omega_k}{\omega_{k+1}}$. Проаналізуємо ε_k.

Нехай обидва i_k, j_k непарні і без обмеження загальності, нехай $i_k > j_k$. Тоді

$$\varepsilon_k := \frac{\omega_k}{\omega_{k+1}} = \frac{i_k + j_k}{\frac{|i_k - j_k|}{2} + \min\{i_k, j_k\}} = \frac{2(i_k + j_k)}{i_k - j_k + 2j_k} = 2$$

Якщо обидва числа парні, то з пари (i_k, j_k) ми отримуємо $(\frac{i_k}{2}, \frac{j_k}{2})$, то легко бачити, що $\varepsilon_k = 2$.

Якщо ж лише одне з чисел парне, то без обмеження загальності з пари (i_k, j_k) ми отримуємо пару $(\frac{i_k}{2}, j_k)$, тому

$$\varepsilon_k = \frac{i_k + j_k}{\frac{i_k}{2} + j_k} > \frac{i_k + j_k}{i_k + j_k} = 1,\ \varepsilon_k < \frac{i_k + j_k}{\frac{i_k}{2} + \frac{j_k}{2}} = 2$$

То бачимо, що $\forall k \in \mathbb{N}, \varepsilon_k \in (1, 2]$.

В найгіршому для алгоритму випадку, НСД дорівнює 1, що відповідає $\omega_N = 1$. Подивимось, які обмеження ми маємо на N відносно вхідних параметрів (a, b). Помітимо, що:

$$1 = \frac{i_{N-1} + j_{N-1}}{\varepsilon_{N-1}} = \frac{i_{N-2} + j_{N-2}}{\varepsilon_{N-1}\varepsilon_{N-2}} = \frac{a + b}{\prod_{j=0}^{N-1} \varepsilon_j}$$

Оскільки $\forall j \in \mathbb{Z}_+ \ \varepsilon_j \in (1, 2]$, то $\prod_{j=0}^{N-1} \varepsilon_j \in (1, 2^{N-1}]$. Бачимо, що

$$\prod_{j=1}^{N-1} \varepsilon_j = a + b$$

В найкращому випадку $\prod_{j=1}^{N-1} \varepsilon_j = 2^{N-1} = a + b \to N = \log_2(a + b) + 1 = \Omega(\log n)$, але в найгіршому випадку $N = a + b$, тобто $\mathcal{O}(n)$, де $n = a + b$.

Тобто в найгіршому випадку маємо $\mathcal{O}(n)$ складність, в найліпшому $\Omega(\log n)$, де $n = a + b$.