Wiktor Kuchta

3/1

---

procedure GCD(a, b)
    procedure ITER(a, b, m)
        if $a > b$ **then**
            return ITER(b, a, m)
        else if $a = 0$ **then**
            return mb
        else if even a **and** even b **then**
            return ITER($a/2$, $b/2$, $2m$)
        else if even a **and** odd b **then**
            return ITER($a/2$, b, m)
        else if odd a **and** even b **then**
            return ITER(a, $b/2$, m)
        else if odd a **and** odd b **then**
            return ITER(a, $(b - a)/2$, m)
        end if
    end procedure
    return ITER(a, b, 1)
end procedure

W każdym wywołaniu (poza zamianą argumentów), jeden z argumentów jest co najmniej dwa razy mniejszy niż wcześniej. Zatem algorytm wykonuje $O(\lg a + \lg b)$ wykonań rekurencyjnych.

Przy jednorodnym kryterium kosztów operacje w każdym wywołaniu zajmują $O(1)$ czasu, a przy logarytmicznym $O(\lg a + \lg b)$.

Zakładając optymalizację wywołań ogonowych, przy jednorodnym kryterium kosztów algorytm potrzebuje $O(1)$ pamięci, a przy logarytmicznym $O(\lg a + \lg b)$. Trzymanie akumulatora m nie zwiększa zużycia pamięci, bo początkowo zajmuje 1 bit, a zwiększamy liczbę zajmowanych bitów o 1 kiedy zmniejszamy liczbę bitów zajmowanych przez a i b o 1.

Algorytm Euklidesa ma takie samy koszty każdej iteracji i potrzebuje tyle samo pamięci, ale wykonuje tylko $O(\lg \min\{a, b\})$ iteracji.

3/2

Liczba porównań algorytmu *MaxMin* dla zbioru n-elementowego jest równa

$$c(n) = \begin{cases} 0 & n < 2 \\ 1 & n = 2 \\ c\left(\lfloor \frac{n}{2} \rfloor\right) + c\left(\lceil \frac{n}{2} \rceil\right) + 2 & n > 2 \end{cases}.$$

Pokażemy indukcyjnie, że

$$\left\lceil \tfrac{3}{2}n \right\rceil - 2 = c(n) \iff \exists k.\, n \in \{2^k - 1, 2^k, 2^k + 1\}.$$

Można sprawdzić, że to prawda dla $n \leqslant 5$. Załóżmy, że to prawda dla liczb mniejszych od n.

1. Jeśli $n = 2^k + \sigma$, gdzie $\sigma \in \{-1, 0, 1\}$, to $c(n) = c(2^{k-1}) + c(2^{k-1} + \sigma) + 2$. Z założenia indukcyjnego jest to równe $\left\lceil \frac{3}{2}2^{k-1} \right\rceil + \left\lceil \frac{3}{2}(2^{k-1} + \sigma) \right\rceil - 2 = \left\lceil \frac{3}{2}(2^k + \sigma) \right\rceil - 2$.
2. Jeśli $n = 2^k \pm 2$, to $c(n) = 2c(2^{k-1} \pm 1) + 2 = 2\left\lceil \frac{3}{2}(2^{k-1} \pm 1) \right\rceil - 2 = \left\lceil \frac{3}{2}2^k \right\rceil + 2\left\lceil \pm\frac{3}{2} \right\rceil - 2 = \left\lceil \frac{3}{2}(2^k \pm 2) \right\rceil - 1$.
3. Jeśli n nie jest żadnym z powyższych, to jedno z $c\left(\lfloor \frac{n}{2} \rfloor\right), c\left(\lceil \frac{n}{2} \rceil\right)$ jest nieoptymalne, tzn. $c(n) > \left\lceil \frac{3}{2}\lfloor \frac{n}{2} \rfloor - 2 \right\rceil + \left\lceil \frac{3}{2}\lceil \frac{n}{2} \rceil - 2 \right\rceil + 2 \geqslant \left\lceil \frac{3}{2}n - 2 \right\rceil$.

Różnica $c(n)$ i $\left\lceil \frac{3}{2}n \right\rceil$ jest nieograniczona.

Dla n parzystych, możemy podzielić zbiór na pary i porównać każdą parę, otrzymując po $n/2$ potencjalnych minimów i maksimów. W tych grupach znajdujemy w $n/2 - 1$ porównań odpowiednio minimum i maksimum. W sumie skorzystaliśmy z $n/2 + 2(n/2 - 1) = 3n/2 - 2$ porównań. Jeśli mamy dodatkowo jeszcze jeden element, to musimy wykonać jeszcze dwa porównania, więc ogólnie ten algorytm wykonuje $\lceil \frac{3n}{2} \rceil - 2$ porównań.

3/3

Mamy dwie otoczki wypukłe rozdzielone pionową kreską. Musimy znaleźć górną i dolną prostą styczną do obu otoczek. Takie proste styczne przechodzą przez wierzchołki otoczek, więc będziemy rozpatrywać odcinek łączący wierzchołki obu otoczek i zmieniać końce tego przecinka, aż odcinek będzie leżał na odpowiedniej prostej.

Zajmiemy się górną styczną, dla dolnej jest symetrycznie. Zaczniemy z odcinkiem przechodzącym przez punkt najbardziej na prawo lewej otoczki i punkt najbardziej na lewo prawej otoczki. Aby sprawdzić, czy prosta, na której leży ten odcinek, przecina prawą figurę i powinniśmy jej prawy koniec przesunąć do wierzchołka wyżej, wystarczy sprawdzić czy te wierzchołki (obecny i potencjalny nowy koniec) prawej figury leżą po odpowiedniej stronie prostej. Symetrycznie dla lewej figury.

Zatem będziemy przesuwać końce odcinka do góry, aż ta prosta nie będzie przecinać żadnej z figur. Wykonamy takich porównań i przesunięć co najwyżej tyle, ile jest wierzchołków na otoczkach.