Lista 3 z AiSD

Zadanie 5

Kamil Banaś 308262

5 maja 2020

1 Treść

Algorytm Euklidesa wyznacza $\gcd(x, y)$ w czasie $O(\log \min(x, y))$. Skonstruuj algorytm, który wyznacza $\gcd(x_1, ..., x_n)$ w czasie $O(n + \log \min(x_1, ..., x_n))$

2 Teoria

Na początku przyjrzyjmy się złożoności czasowej algorytmu Euklidesa. Widzimy, że czas wykonania się algorytmu zależy od końcowego $\gcd(x, y)$, czego nie zawieramy w aktualnej złożoności czasowej. Wobec tego udowodnijmy następujące lematy:

2.1 Lemat 1

Jeżeli $x \equiv y \bmod z$, to $dx \equiv dy \bmod dz$ dla $x, y, z, d \in \mathbb{N}$.
Dowód: Z pierwszej kongruencji widzimy, że $x = qz + y,\ q \in \mathbb{N}$, stąd $y < z \Rightarrow dy < dz$. Zatem $dx = dqz + dy$, czyli $dqz + dy \equiv dy \bmod dz$

2.2 Lemat 2

Algorytm Euklidesa wykonuje tyle samo kroków dla pary liczb (x, y) jak i dla pary liczb (dx, dy), $x, y, d \in \mathbb{N}$.
Dowód:Lemat udowodnimy na podstawie indukcji względem kroków Algorytmu Euklidesa. W danym kroku Algorytm Euklidesa przyjmuje parę liczb (x, y), gdzie BSO $x > y$, i zwraca parę liczb $(x \bmod y, y)$. Powiemy, że pary (x, y) i (a, b) są *dobre*,
jeżeli $a = dx, b = dy$.
Teza indukcji: Niezmiennikiem w obu algorytmach jest fakt, że po dowolnej ilości kroków Algorytmu Euklidesa dla obu par powstałe pary są *dobre*.
Baza indukcji: W pierwszym kroku dla pary (x, y) dostajemy parę $(x \bmod y, y)$. Natomiast dla pary (dx, dy) dostajemy parę $(dx \bmod dy, dy)$. Z lematu 1 otrzymujemy, że $d * (x \bmod y) = dx \bmod dy$. Zatem otrzymane pary są *dobre*.
Krok indukcyjny: Zakładamy, że po n-tym kroku mamy pary (x_n, y_n) i (dx_n, dy_n), które są *dobre*. Analogicznie jak w bazie indukcji pokazujemy, że po $n + 1$-szym kroku mamy pary $(x_n \bmod y_n, y_n)$ i $(dx_n \bmod dy_n, dy_n)$, gdzie $d * (x_n \bmod y_n) = dx_n \bmod dy_n$. Czyli ponownie pary otrzymane są *dobre*

Wniosek: Z powyższej indukcji wynika, że po dowolnym kroku Algorytmu Euklidesa pary są *dobre*. Algorytm kończy się, jak mniejszy element w parze staje się równy 0. Jeżeli w jednej parze otrzymujemy 0, to z definicji par *dobrych* w drugiej parze też będzie 0.

2.3 Lemat 3

Algorytm Euklidesa wyznacza $\gcd(x, y)$ w czasie $O(\log(\min(x, y)/\gcd(x, y)))$.
Dowód: Niech $x' = x/\gcd(x, y)$, a $y' = y/\gcd(x, y)$. Z treści zadania wiemy, że Algorytm Euklidesa wyznacza $\gcd(x', y')$ w czasie $O(\log(\min(x', y')))$. Z lematu 2 ponieważ $\gcd(x, y) \in \mathbb{N}$, to Algorytm Euklidesa wyznacza $\gcd(x, y)$ w takim samym czasie co $\gcd(x', y')$, czyli w czasie $O(\log(\min(x', y'))) = O(\log(\min(x, y)/\gcd(x, y)))$.

3 Rozwiązanie

3.1 Algorytm

Algorytm rozwiązania wygląda następująco:

swap$(x_1, \min(x_1, ..., x_n))$
for $i = 2; i \leqslant n; i \leftarrow i + 1$ **do**
  $x_1 = \gcd(x_1, x_i)$
end for
return x_1

Widzimy, że to jest, z wyjątkiem zamiany dwóch elementów, elementarny sposób obliczenia gcd wielu liczb.

3.2 Uzasadnienie złożoności czasowej

Niech $(x_1, ..., x_n)$ to będzie ciąg liczb, których gcd musimy policzyć.
Weźmy $x_1 = min(x_1, ..., x_n)$, co możemy osiągnąć iterując przez ciąg, żeby znaleźć wartość minimum ciągu i zamienić ją miejscami z wartością x_1.
Policzymy gcd ciągu w elementarny sposób:
Przyjmiemy $D = x_1$ i przeiterujemy przez ciąg ponownie licząc dla $i = 2, ..., n$

$$D := \gcd(D, x_i).$$

Policzymy teraz, ile łącznie zajmują czasu obliczenia gcd. Niech D_i to liczba równa D po i-tej iteracji algorytmu, czyli:

$$D_1 = x_1,$$

$$D_i = \gcd(D_{i-1}, x_i).$$

W ciągu i-tej iteracji obliczamy $\gcd(D_{i-1}, x_i)$ otrzymując w wyniku D_i. Wobec tego złożoność czasowa tej operacji jest równa z lematu 3 $O(\log(\min(D_{i-1}, x_i)/D_i))$.
Oczywiście $D_{i-1} \leqslant x_1 \leqslant x_i$, stąd mamy ostateczną złożoność

$$O(\log(D_{i-1}/D_i)) = O(\log\, D_{i-1} - \log\, D_i).$$

Zostało nam policzenie sumy czasów wszystkich iteracji gcd:

$$\sum_{i=2}^{n}(\log\, D_{i-1} - \log\, D_i) = \log\, D_1 - \log\, D_n \leqslant \log\, D_1 = \log\, x_1.$$

Na ostateczną złożonością czasową wpływa iteracja po ciągu podczas szukania minimum, iteracja po ciągu podczas liczenia gcd oraz samo liczenie gcd. Zauważmy też, że samo liczenie gcd w każdej iteracji zajmuje co najmniej czas stały, wobec czego liczenie gcd wszystkich elementów zajmuje $O(n + \log\, x_1)$. Biorąc to wszystko pod uwagę złożonością czasową algorytmu jest $O(n + n + n + \log\, x_1) = O(n + min(x_1, ..., x_n))$.

$\square$