Zajęcia A-6: “Funkcje / sprawdzanie pierwszości”

Cel zajęć i efekty uczenia

Główne cele zajęć / materiał do opanowania:

- Implementacja “pierwiastkowego” algorytmu sprawdzania pierwszości
- Zdobycie pierwszych intuicji w kwestii notacji O() i asymptotycznej złożoności algorytmu
- Pojęcie funkcji (podprogramu) w języku programowania, implementacja funkcji w C++

Dodatkowe cele:

- Utrwalenie konstrukcji pętli *for* w języku C++
- Intuicja “małych” i “dużych” dzielników, znajdowanie wszystkich dzielników w czasie pierwiastkowym
- Elegancja w projektowaniu algorytmu, unikanie powtarzania kodu, szczególnie “copy-paste”
- Dbałość o szczegóły podczas pisania programu

Zadania do rozwiązania na sprawdzarce

Zielone butelki
Wypisać wiersz o butelkach:

“10 green bottles standing on the wall (…)
and if one green bottle should accidentally fall,
there’d be 9 green bottles standing on the wall”

w wersji dla 100 000 butelek. Zamiast każdej liczby pierwszej należy wypisać BUZZ.

[opcjonalnie] ***Faktoryzacja***
Mając daną liczbę całkowitą N, wypisać jej rozkład na czynniki pierwsze

Plan zajęć

Szacunkowy czas trwania: 4 godziny lekcyjne.

1. Jakie byłoby rozwiązanie zadania, gdyby chodziło tylko o wypisanie wiersza, bez warunku z liczbami pierwszymi?
    - *Warto przypomnieć konstrukcję pętli for i poświęcić chwilę na przypomnienie, jak ją implementować, aby działała “w dół” (od N do 1).*
2. Przypomnienie pojęcia liczby pierwszej
3. Jak sprawdzać pierwszość zadanej liczby N: algorytm “podziel przez wszystkie dzielniki mniejsze niż N”
4. Poprawka algorytmu: dzielenie przez liczby mniejsze od N/2, i/lub przez liczby nieparzyste

- *Warto od razu zastanawiać się nad złożonością na konkretnym, dużym N (~10^9), aby zilustrować niewielki zysk na złożoności.*

5. Intuicja złożoności asymptotycznej i notacji O()
   - *To dobry moment do zilustrowania, czemu poprawki N -> N/2 są z punktu widzenia algorytmiki mniej ważne: ten sam efekt można osiągnąć biorąc szybszy komputer, albo lepszy kompilator języka.*
   - *Zalecam w tym momencie zaprezentować zapis O(N) w znaczeniu "N * jakaś liczba" - przy czym "jakaś liczba" jest stałą, którą możemy w każdej chwili wyznaczyć, ale nie jest nam to na razie potrzebne.*
   - *Warto jednak podkreślić, że w "zawodowym" programowaniu poprawki stałej mogą się okazać bardzo ważne.*
6. Małe i duże dzielniki
   - *Proponuję rozpisać dzielniki liczby 36 (albo np. 144) w parach: 36 = $1 \cdot 36 = 2 \cdot 18 = 3 \cdot 12 = \ldots$ - z tej postaci widać, że połowa dzielników jest "mała" (mniejsza niż sqrt(N), a połowa "duża" (większa niż sqrt(N)).*
7. Modyfikacja do algorytmu "pierwiastkowego"
8. Włączenie algorytmu na pierwszość do zadania - jak uniknąć wielokrotnego przepisywania?
9. Pojęcie podprogramu (funkcji) i implementacja w C++
   - *Zaczynamy od prostej funkcji "podnieś do kwadratu", potem np. "f(x,y) = x+y-1", aby dojść do funkcji "f(x) = 0 jeśli x nie jest pierwsza, f(x) = 1 jeśli x jest pierwsza".*
   - *Uczulamy uczniów na fakt, że napotykając wywołanie funkcji, program zapamiętuje, gdzie był, przeskakuje na chwilę do kodu funkcji, a potem wraca, przywracając wszystko do stanu sprzed wywołania.*
10. [opcjonalnie] Algorytm faktoryzacji - zadanie *Faktoryzacja*
    - *W najprostszej wersji, algorytm po prostu dzieli N przez kolejne liczby naturalne - każda liczba, przez którą uda się podzielić, musi być liczbą pierwszą (nie uda się podzielić np. przez 4, bo wcześniej już dzieliliśmy przez 2). Dzielenie wykonuje się do pierwiastka z N, ostatnia pozostała liczba jest ostatnim dzielnikiem pierwszym N.*