Zajęcia B-2: “Wstęp do rekursji / Szybkie potęgowanie”

Cel zajęć i efekty uczenia

Główne cele zajęć / materiał do opanowania:
- Wprowadzenie pojęcia rekursji i funkcji rekurencyjnej
- Opanowanie i implementacja algorytmu szybkiego potęgowania

Dodatkowe cele:
- Intuicje dotyczące arytmetyki modularnej
- Zrozumienie, jak działają funkcje w programie (stos wywołań)

Zadania do rozwiązania na sprawdzarce

Szybkie potęgowanie

Mając dane liczby a, k i p, obliczyć a^k modulo p.

Plan zajęć

Szacunkowy czas trwania: 2 godziny lekcyjne.
1. Definicja silni, obliczanie metodą iteracyjną
   - *Prosta pętla, uczniowie dochodzą do niej sami*
2. Wyjaśnienie działanie wywołania funkcji (odłóż na stos wywołań, przejdź do funkcji, wróć)
3. Wywołanie rekurencyjne w silni - jak zadziała?
   - *Należy dokładnie opisać działanie - każdy krok wykonany przez program - w wywołaniu np. silnia(4)*
4. Kiedy (i jak) w ogólności działa rekursja
   - *Rekurencja wymaga trzech warunków do działania:*
     - *wywołanie istotnie następuje na mniejszych danych (przejście z silnia(4) do silnia(3), a nie np. do silnia(6)).*
     - *wiemy, jak z wyniku dla mniejszej liczby odtworzyć wynik dla większej (przejście silnia(3) -> silnia(4) jest łatwe).*
     - *znamy początek rekursji (umiemy obliczyć silnia(0) bez wywołania rekurencyjnego).*
5. Implementacja iteracyjna, arytmetyka modularna
   - *Iteracyjna wersja potęgowania jest bardzo prosta dla uczniów, szczególnie znających już silnię*
   - *Reszta z dzielenia - dlaczego nie można obliczyć a^k, a potem wyciągnąć reszty*
   - *Typowe pytanie uczniów: dlaczego nie zadziała funkcja pow(...)?*
   - *Konieczność wyjaśnienia/przypomnienia, jak działa reszta z dzielenia - można brać resztę po każdej operacji, i otrzyma się ten sam wynik*
6. Implementacja rekurencyjna i modyfikacja do wersji “szybkiej” potęgowania
   - *Przepisujemy analogicznie do silni, rysujemy łańcuch wywołań np. dla 2^7*

- *Zauważamy, że dla liczb parzystych (i tylko dla nich) wywołanie a*potega(a,k-1) można zamienić na potega(a,k/2) podniesione do kwadratu*
- *Rozrysowujemy łańcuch wywołań rekurencyjnych np. dla 2^{50}*
- *Argumentujemy, że co drugie wywołanie zmniejsza k o połowę, więc całkowita liczba wywołań nie przekroczy log k.*