1 Wprowadzenie

Standardowe potęgowanie dla wyrażenia a^n potrzebuje aż n-1 mnożeń. Istnieje jednak o wiele wydajniejszy sposób na podnoszenie wyrażenia do potęgi naturalnej, zwłaszcza dla dużych wykładników. Nazywa się on potęgowanie szybkie. Algorytm pozwala na zrealizowanie tego zadania wykonując maksymalnie 2*log2(n) mnożeń. Dla dużych wykładników oszczędność jest więc ogromna.

Dla przykładu obliczmy wartość wyrażenia 2^{10}:

$2^{10} = 2*2*2*2*2*2*2*2*2*2 = 1024$

Wykonaliśmy aż 9 operacji mnożenia.

Potęgę 2^{10} można jednak obliczyć w inny sposób:

$2^{10} = (2^5)^2 = (2*2^4)^2 = (2*(2^2)^2)^2 = (2*(2*2)^2)^2$

W tym przypadku otrzymujemy tylko 4 operacje mnożenia. W czym tkwi zatem tajemnica? Cały geniusz tej metody kryje się w wykładniku, a dokładniej w jego postaci binarnej, w tym przypadku $10 = 1010_2$.

2 Opis działania

Działanie metody opiera się na binarnym systemie pozycyjnym oraz na regule dodawania wykładników podczas mnożenia potęg o tych samych podstawach. Zasada jest następująca:

1) przypisujemy wynik = 1
2) dla kolejnych od prawej bitów:
   a) jeśli bit jest równy 1, przypisujemy wynik = wynik * a
   b) przypisujemy a = a * a
3) wypisujemy wynik

3 Specyfikacja

Dane:
- liczba rzeczywista: **a**>0 (podstawa potęgi)
- liczba naturalna: **n** (wykładnik potęgi)

Wynik:
liczba rzeczywista będąca wynikiem podniesienia liczby a do potęgi n-tej: wynik

4 Przykładowy kod w języku Python3 (funkcja iteracyjna)

```
    # iteracyjny algorytm szybkiego podnoszenia liczby do naturalnej potęgi
def szybkie_potegowanie_iteracja(a,n):
    wynik = 1 # nadajemy zmiennej przechowującej wynik wartość początkową
    while n>0: # pętla wykonuj liczbę iteracji równa ilości cyfr w zapisie>
        if n%2: # sprawdzamy, czy wartość bitu równa się jeden
            wynik*=a # jeżeli tak to mnożymy wynik przez podstawę
        a*=a # podnosimy podstawę do potęgi
        n//=2 # dzielimy całkowicie wykładnik przez dwa
    return wynik # zwracamy wynik
```

5 Przykładowy kod w języku Python3 (funkcja rekurencyjna)

```
# rekurency algorytm szybkiego podnoszenia liczby do potęgi naturalnej
def szybkie_potegowanie_rekurencja(a,n):
        if n==0: # jeżeli wykładnik jest równy zero, to zwracamy 1
                return 1
        elif n==1: # jeżeli wykładnik jest równy  jeden, to zwracamy podstawę
                return a
        elif n%2: # jeżeli wartość bitu jest równa jeden, to należy końcowy wynik przemnożyć przez podstawę
        return szybkie_potegowanie_rekurencja(a*a,n//2)*a
# (return szybkie_potegowanie_rekurencja(a,n//2)*szybkie_potegowanie_rekurencja(a,n//2)*a)
        return szybkie_potegowanie_rekurencja(a*a,n//2)
# (return szybkie_potegowanie_rekurencja(a,n//2)*szybkie_potegowanie_rekurencja(a,n//2))
```

6 Zastosowanie

Szybkie podnoszenie do potęgi w praktyce stosuje się do obliczania reszty z dzielenia potęgi przez ustaloną liczbę. Używa się go np. w algorytmach szyfru RSA.