Najważniejszy fragment kodu źródłowego

```
1.  def towerOfHanoi(n, S, D, H):
2.      if n == 1:
3.          D.append(S.pop())
4.          return
5.      Else:
6.          towerOfHanoi(n-1, S, H, D):
7.          D.append(S.pop())
8.          towerOfHanoi(n-1, H, D, S):
```

Uzasadnienie

Wybrałem powyższy fragment kodu, ponieważ w tym laboratorium sprawił mi on najwięcej trudności. Spędziłem wiele czasu próbując napisać tę funkcję i jednocześnie analizując zagadnienie wierzy Hanoi. Jest to doskonały przykład rekurencji, ponieważ rozwiązanie rekurencyjne jest tutaj bardziej intuicyjne od rozwiązania iteracyjnego. Myślę, że z tego powodu poczyniłem duży krok na drodze do zrozumienia rekurencji próbując stworzyć ten konkretny fragment kodu.

Podsumowanie

Obie wersje, zarówno rekurencyjna jak i iteracyjna, są optymalne pod względem ilości ruchów, które trzeba wykonać. Jest to zależność potęgowa opisana wzorem $2^n - 1$, gdzie n jest liczbą krążków. Porównując działanie algorytmów dla wielu przykładów można szybko przekonać się, że oba prowadzą do identycznych rezultatów, wykonując po drodze dokładnie te same ruchy.

Warto w tym miejscu zauważyć, że w zależności od tego czy liczba krążków jest parzysta czy nieparzysta dla słupków A,B,C pierwszym ruchem będzie przełożenie krążka numer 1 na słupek B, bądź na słupek C, od czego później zależą wszystkie pozostałe ruchy. W przypadku wersji rekurencyjnej rozstrzygnięcie tego problemu wynika bezpośrednio z implementacji algorytmu. Jednak w wersji iteracyjnej należy manualnie dokonać tego wyboru poprzez podanie argumentów do funkcji w odpowiedni sposób.

Pomimo, że oba algorytmy wykonują taką samą liczbę ruchów, różnica w czasie jest znacząca. Im większa jest liczba krążków, tym widoczniejsza staje się przewaga algorytmu rekurencyjnego nad iteracyjnym. Załączone do mojej pracy programy timerR oraz timerI mierzą czasy wykonania 100 powtórzeń algorytmu dla liczby krążków od 10 do 19. Zebrałem te dane i przedstawiłem je na poniższym wykresie.

Kolor czerwony- algorytm iteracyjny
Kolor niebieski- algorytm rekurencyjny