Algorytmy i struktury danych, Teleinformatyka, I rok

Raport z laboratorium nr: a02

Imię i nazwisko studenta: Maciej Klimek

nr indeksu: 414836

1. W pole poniżej wklej najważniejszy (według Ciebie) fragment kodu źródłowego z zajęć (maksymalnie 15 linii).

```
1. def moveDisk(stack1, stack2):
2.     if stack1.lastIndex == -1 or (stack1.tbl[stack1.lastIndex] >
   stack2.tbl[stack2.lastIndex] and stack2.tbl[stack2.lastIndex] != 0):
3.         moveDisk(stack2, stack1)
4.         return
5.     #print(f"Przekładam krążek {stack1.tbl[stack1.lastIndex]} z drążka
   {stack1.name} na drążek {stack2.name}")
6.     stack2.lastIndex += 1
7.     stack2.tbl[stack2.lastIndex] = stack1.tbl[stack1.lastIndex]
8.     stack1.tbl[stack1.lastIndex] = 0
9.     stack1.lastIndex -= 1
```

Uzasadnij swój wybór.

Funkcja ta jest podstawowym elementem algorytmu w wersji iteratywnej. Pierwotnie w moim pomyśle stosy, których tabela elementów była pusta miały zmienną lastIndex = 0. Stworzyło to jednak problemy w porównywaniu elementów dwóch stosów w tej właśnie funkcji (w sytuacji gdy stos miał tylko jeden element). Rozwiązaniem, które zaimplementowałem jest zamienienie pierwotnej wartości lastIndex na -1, w ten sposób można jasno zidentyfikowac kiedy stos jest pusty i nie występuje konflikt przy porównywaniu pierwszego elementu tablicy. Kluczowym elementem jest też pierwsze kilka linii funkcji - rozstrzyganie w którą stronę mamy wykonać dany ruch (z A do B, czy może z B do A).

2. Podsumuj wyniki uzyskane podczas wykonywania ćwiczenia. Co ciekawego zauważyłeś? Czego się nauczyłeś? Jeśli instrukcja zawierała pytania, odpowiedz na nie. Do sprawozdania możesz dodać wykresy jeśli jest taka potrzeba.

Obie wersje algorytmu są optymalne – wykonują taką samą ilość ruchów dla danej ilośći krążków. W kwesti implementacji algorytm rekurencyjny sprawia wrażenie zdecydowanie łatwiejszego. W tym przypadku koncept rekurencji jest również bardzo intuicyjny. Algorytm iteracyjny prezentuje wiele problemów do rozstrzygnięcia, z którymi przy rekurencji nie było do czynienia(np. Sprawdzanie w którą strone będziemy wykonywać ruch). Z pomiarów czasowych wynika że algorytm iteracyjny jest na ogół wolniejszy niż rekurencyjny. Przykładowo dla 100 pomiarów, każdy z nich z 15 krążkami:

- Średni czas rekurencyjnie: 0.004423425197601318
- Średni czas iteracyjnie: 0.01743739366531372