Problemy Rekurencyjne

1 (Wieże z Hanoi, Eduard Lucas, 1883)**.** Mamy wieżę złożoną z n krążków, $n \in \mathbb{N}_0$, ułożonych jeden na drugim i nadzianych na jeden z 3 prętów (oznaczmy go przez A) malejącymi średnicami. Zadanie polega na przeniesieniu całej wieży krążków na jeden z pozostałych prętów (oznaczmy go przez B), przy czym w każdym ruchu można brać tylko jeden krążek i nie wolno położyć większego krążka na mniejszym.

a) Ile co najmniej ruchów trzeba wykonać, aby przenieść wieżę?

b) Napisz program wyznaczający kolejne przełożenia dla klasycznego problemu wieży Hanoi.

c) Zapisz powyższy algorytm w postaci programu w dowolnym języku programowania. Program czyta ze standardowego wejścia nieujemną liczbę całkowitą n (liczba krążków) i wypisuje w kolejnych wierszach standardowego wyjścia kolejne przełożenia.

Przykład.

Dla danych wejściowych
3
poprawną odpowiedzią jest:
(A, B)
(A, C)
(B, C)
(A, B)
(C, A)
(C, B)
(A, B)

2. Znajdź najkrótszą sekwencję ruchów przekładających wieżę złożoną z n krążków z pręta A na pręt B, jeśli bezpośrednie ruchy między A i B są zabronione. Napisz odpowiedni program.

3. Podwójna wieża z Hanoi składa się z $2n$ krążków n różnych rozmiarów, po 2 krążki każdego rozmiaru.

a) Ile ruchów trzeba co najmniej wykonać, aby przenieść wieżę z jednego pręta na drugi, jeśli krążki równej wielkości są nierozróżnialne?

b) Co będzie, gdy zażądamy odtworzenia oryginalnego ułożenia krążków?

4 (Problem Flawiusza)**.** Załóżmy, że mamy n ludzi ponumerowanych od 1 do n ustawionych w kręgu. „Eliminować" będziemy co drugą osobę, aż pozostanie tylko jeden człowiek. Nasz problem polega na wyliczeniu przeżywającego numeru $J(n)$, czyli znalezieniu wzoru ogólnego lub rekurencyjnego dla funkcji J. Napisz program, który zwraca stosowną wartość (istnieje dość proste rozwiązanie rekurencyjne, ale trzeba na nie wpaść).

Np. dla $n = 10$ porządek eliminowania jest następujący

$$2,\ 4,\ 6,\ 8,\ 10,\ 3,\ 7,\ 1,\ 9,$$

tak więc $J(10) = 5$.

5. Podaj algorytm i napisz program, który opierając się na rekurencji wypisuje wszystkie anagramy podanego słowa.

Wejście. Program czyta ze standardowego wejścia. W pierwszym i jedynym wierszu podane jest dowolne słowo o długości $\leqslant 255$.

Wyjście: Program powinien wypisać w kolejnych wierszach standardowego wyjścia kolejne anagramy badanego słowa wyjście.

Przykład:
Dla danych wejściowych:
abc
poprawną odpowiedzią jest:
abc
acb
bac
bca
cab
cba

6. Napisz rekurencyjną funkcję sumującą pierwszych n składników szeregu

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} \ldots.$$

7. Napisz rekurencyjną funkcję sumującą pierwszych n składników szeregu

$$1 + \frac{1}{2} - \frac{1}{3} + \frac{1}{4} - \frac{1}{5} \ldots.$$

8. Sortowanie przez wstawianie może być wyrażone jako procedura rekurencyjna. Spróbuj napisać odpowiedni program.

9. Zaimplementuj rekurencyjną wersję algorytmu Euklidesa obliczającego największy wspólny dzielnik dwóch liczb naturalnych.

10. Napisz algorytm (z wykorzystaniem pętli while) wyszukiwania binarnego klucza *key* w posortowanej rosnąco liście A. Jeżeli klucz *key* znajduje się w liście A, to algorytm powinien zwrócić taki indeks k, że $A[k] = key$. Jeżeli klucz *key* nie znajduje się w liście A, to algorytm powinien zwrócić `None`.
Jaka jest wg Ciebie złożoność czasowa optymistyczna i pesymistyczna podanego algorytmu?

11. Napisz algorytm (z wykorzystaniem rekurencji) wyszukiwania binarnego klucza *key* w posortowanej rosnąco liście A. Jeżeli klucz *key* znajduje się w liście A, to algorytm powinien zwrócić taki indeks k, że $A[k] = key$. Jeżeli klucz *key* nie znajduje się w liście A, to algorytm powinien zwrócić `None`.
Jaka jest wg Ciebie złożoność czasowa optymistyczna i pesymistyczna podanego algorytmu?

12. Porównaj (na swoim komputerze) czas działania sortowaniem przez scalanie (MergeSort) oraz sortowania szybkiego (QuickSort) dla 10 ciągów losowych, ciągu posortowanego rosnąco oraz ciągu posortowanego malejąco. Korzystamy z kodów podanych na wykładzie. Jako długość ciągu przyjmij $n = 10000$ oraz $n = 50000$.

Napisz swoje własne wersje powyższych algorytmów z wykorzystaniem "wycinania" list w Pythonie. Ponownie porównaj czasy działania tych algorytmów.