*Zadanie domowe*

1. Rekurencja ogonowa jest techniką programowania, która polega na optymalizacji rekurencyjnej funkcji poprzez przekształcenie jej w pętlę. W tradycyjnej rekurencji, każde wywołanie funkcji musi poczekać na zwrócenie wartości przez kolejne wywołanie, zanim może zakończyć swoje działanie. W rekurencji ogonowej, ostatnia operacja wykonywana przez funkcję to wywołanie samej siebie. W wyniku tego, kompilator lub interpreter może zastąpić wywołanie rekurencyjne przez skok do początku funkcji z nowymi parametrami, co oznacza, że kolejne wywołania nie będą umieszczane na stosie. Dzięki temu unikamy przepełnienia stosu.

public class Silnia {

*//Metoda wyliczająca silnię za pomocą zwykłej rekurencji*

public static int SilniaRekurencyjna(int n) {

if (n == 0) return 1;

else return n \* SilniaRekurencyjna(n - 1);

}

*//Metoda pomocnicza dla metody SilniaOgonowa*

private static int SilniaOgonowa(int n, int buf) {

if (n == 0) return buf;

else return SilniaOgonowa(n - 1, n \* buf);

}

*//Metoda wyliczająca silnię za pomocą rekurencji ogonowej*

public static int SilniaOgonowa(int n) {

return silniaOgonowa(n, 1);

}

public static void main(String[] args) {

int n = 5;

System.out.println("Silnia " + n + " wynosi:");

System.out.println("Zwykła rekurencja: " + SilniaRekurencyjna(n));

System.out.println("Rekurencja ogonowa: " + SilniaOgonowa(n));

}

}

1. Metoda "dziel i zwyciężaj" jest jednym z podstawowych algorytmów sortowania, a jej zasadnicza idea polega na podziale zadania na mniejsze podproblemy, rozwiązanie ich rekurencyjnie, a następnie połączeniu wyników w celu uzyskania ostatecznego rozwiązania.

Algorytm Mergesort działa w oparciu o tę metodę i działa w następujący sposób:

1) Podziel tablicę na pół, tworząc dwie mniejsze tablice.

2) Rekurencyjnie posortuj każdą z tych dwóch mniejszych tablic, stosując algorytm Mergesort.

3) Scal te dwie posortowane mniejsze tablice w jedną większą, posortowaną tablicę. To właśnie etap scalania jest kluczowy dla algorytmu Mergesort.

Powtarzaj etapy 1-3, aż cała tablica zostanie posortowana.

1. Złożoność czasowa algorytmu Mergesort wynosi O(n\*log n), co oznacza, że czas sortowania rośnie w sposób logarytmiczny wraz ze wzrostem liczby elementów do posortowania.

Podział listy na dwie połowy ma złożoność O(log n) w czasie, ponieważ dla każdego podziału liczba elementów do posortowania zmniejsza się dwukrotnie.

Sortowanie każdej z połówek ma złożoność O(n), ponieważ musimy porównać każdy element z każdym w celu ustalenia ich względnej kolejności. Łączenie dwóch posortowanych połówek ma złożoność czasową O(n), ponieważ musimy porównać każdy element z jednej listy z każdym elementem z drugiej listy.

1. Algorytm rozwiązujący problem plecakowy opiera się na wykorzystaniu rekurencji i jest często nazywany algorytmem brute-force lub algorytmem siłowym. Polega na wygenerowaniu wszystkich możliwych kombinacji elementów, które mieszczą się w plecaku, a następnie wybraniu tej, która daje największą wartość.

Procedura ta wykorzystuje funkcję f(i, j), która oznacza maksymalną wartość, jaką można uzyskać z i pierwszych elementów, przy czym pojemność plecaka wynosi j. Zauważmy, że wartość funkcji f(n, W) jest właściwym rozwiązaniem problemu plecakowego.

Funkcja ta może być wyrażona przez następujący wzór rekurencyjny:

f(i, j) = 0, gdy i = 0 lub j = 0

f(i, j) = f(i-1, j), gdy w(i) > j

f(i, j) = max(f(i-1, j), v(i) + f(i-1, j-w(i))), gdy w(i) <= j

gdzie w(i) oznacza wagę i-tego elementu, a v(i) jego wartość.

Intuicyjnie, funkcja f(i,j) dla i-tego elementu przyjmuje wartość równą maksimum z dwóch przypadków: albo i-ty element nie jest umieszczony w plecaku i wartość funkcji wynosi wartość funkcji dla i-1 elementów, albo i-ty element jest umieszczony w plecaku i wartość funkcji wynosi sumę wartości tego elementu i wartości, która została osiągnięta dla pozostałej pojemności plecaka (j-w(i)), biorąc pod uwagę tylko i-1 pierwszych elementów.

Ostatecznym wynikiem będzie wartość f(n,W), czyli maksymalna wartość, jaką można uzyskać przy danej pojemności plecaka.

Przykład:

Poniżej są przedstawione kolejne kroki wykonywane dla elementów: 8, 7, 6, 5 do plecaka 20

8, waga docelowa = 20, 8 to za mało

8, 7 waga docelowa = 12, 7 to za mało

8, 7, 6 waga docelowa = 5, 6 to za dużo

8, 7, 5 waga docelowa = 5, 5 pasuje, Koniec

1. Problem Wież Hanoi to klasyczny problem matematyczny polegający na przeniesieniu stosu krążków z jednego kołka na drugi, zgodnie z określonymi zasadami, za pomocą trzeciego kołka jako tymczasowego magazynu. Algorytm rozwiązujący ten problem może być opisany w sposób rekurencyjny.

Ustawienie początkowe:

Początkowo wszystkie krążki znajdują się na pierwszym kołku (źródłowym) ustawione od największego do najmniejszego, na pozostałych kołach nie ma krążków.

Rekurencyjna procedura:

Algorytm rozwiązywania problemu Wież Hanoi polega na przesuwaniu krążków z jednego kołka na drugi zgodnie z następującymi zasadami:

-tylko jeden krążek można przesunąć w danym momencie,

-większy krążek nie może być położony na mniejszym.

Procedura rekurencyjna jest następująca:

Przemieszczenie k pierwszych krążków z kołka A (źródłowego) na kołek B (docelowy) przy użyciu kołka C (tymczasowy) jako magazyn.

Przemieszczenie pozostałych n - k krążków z kołka A na kołek C przy użyciu kołka B jako magazyn, używając tej samej procedury rekurencyjnej.

Przeniesienie k krążków z kołka B na kołek C przy użyciu kołka A jako magazyn, również przy użyciu tej samej procedury rekurencyjnej.

Procedura kończy się, gdy wszystkie krążki są przeniesione z kołka A na kołek C.

