Algorytmy sortujące cz. 1

11 listopada 2015

1 Bubble Sort

Algorytm Bubble Sort porządkuje listę poprzez iteracyjne sprawdzenie czy kolejne elementy listy są umieszczone we właściwej kolejności.
Załóżmy, że do sprawdzenia mamy n elementową listę. W trakcie pierwszego sprawdzenia algorytm porówna n-1 par. Po pierwszym sprawdzeniu, na miejscu n znajdzie się największy element w liście. Przy sprawdzeniu drugim wystarczy porównać n-1 elementów (daje to n-2 pary), na końcu tego kroku na miejscach n i n-1 znajdują się właściwe elementy. Ogółem trzeba wykonać n-1 sprawdzeń.
Przykład: lista wejściowa [4,2,5,1]

1. iteracja
   - 1 para (4,2), przestawienie, lista: [2,4,5,1]
   - 2 para (4,5), brak przestawienia, lista: [2,4,5,1]
   - 3 para (5,1), przestawienie, lista: [2,4,1,5]. Jest to ostatnie porównanie w tej iteracji. Na n (4) miejscu znajduje się największy element w liście
2. iteracja
   - 1 para (2,4), brak przestawienia, lista: [2,4,1,5]
   - 2 para (4,1), przestawienie, lista: [2,1,4,5]. Jest to ostatnie porównanie w tej iteracji. Na n (4) i n-1 (3) miejscu znajdują się właściwe elementy
3. iteracja
   - 1 para (2,1), przestawienie, lista: [1,2,4,5]. Jest to ostatnie porównanie w ostatniej iteracji.

2 Selection Sort

W tym algorytmie w każdej iteracji znajdujemy największy nieuporządkowany element listy i ustawiamy go w odpowiednim miejscu. W 1 iteracji największy element listy ustawiany jest na miejscu n. W 2 iteracji największy nieuporządkowany element trafia na miejsce n-1.
Przykład: lista wejściowa [4,2,5,1]

1. iteracja, największa jest 5, przestawienie [4,2,1,5]
2. iteracja, największa jest 4, przestawienie [1,2,4,5]
3. iteracja, największa jest 2, koniec procedury

3 Insertion Sort

Algorytm ten działa w nieco inny sposób. Dla listy n elementowej zakładamy, że element 1 stanowi posortowaną listę. Kolejny element (2), chcemy dodać do tej listy w odpowiednie miejsce. Następnie dodajemy element 3 do listy już 2 elementowej.
Przykład: lista wejściowa [4,2,5,1]

1. iteracja, listę 1 elementową stanowi 4, kolejny element to 2, umieszczamy go przed 4, lista ma postać (w nawiasach zaznaczono już przesortowane elementy) [(2,4),5,1]

2. iteracja, do listy (2,4) dodajemy 5, [(2,4,5),1]

3. iteracja, do listy (2,4,5) dodajemy 1 otrzymujemy [(1,2,4,5)].

Dodanie elementu do listy odbywa się w określony sposób. Dla przykładu omówimy przypadek dodania 1 do listy listy (2,4,5). Element który dodajemy, czyli 1, zapamiętujemy w pamięci, po czym dokonujemy odpowiedniego przesunięcia elementów już posortowanych:

1. krok: (2,4,5),1; ponieważ $5 > 1$ przesuwany 5 na miejsce 1, po tej operacji 2,4,-,5 (- oznacza pusty element)

2. krok: 2,4,-,5; ponieważ $4 > 1$ przesuwamy 4 na puste miejsce, po tej operacji 2,-,4,5

3. krok: 2,-,4,5; ponieważ $2 > 1$ przesuwamy 2 na puste miejsce i wstawiamy 1 na miejsce 2, po tej operacji 1,2,4,5

4 Zadania i punktacja

Parametrem wejściowym do programu jest liczba określająca długość listy wejściowej n (do 10000). Program powinien wypełnić listę losowymi elementami z przedziału 0-10000. Wejściowa i wyjściowa lista powinna ma być zapisywana do pliku.

- Implementacja algorytmu Bubble Sort wraz ze zliczeniem liczby dokonywanych przestawień 2 punkty
- Implementacja algorytmu Selection Sort wraz ze zliczeniem liczby dokonywanych przestawień 2 punkty
- Implementacja algorytmu Insertion Sort wraz ze zliczeniem dokonywanych operacji przesunięć i przestawień 2 punkty
- Wykonanie wykresu ilości operacji dla powyższych algorytmów w funkcji długości wejściowej listy o rozmiarach n= 10,50, 100, 500, 1000, 5000,10000 - 2 punkty
- Dołączenie do programu funkcji mierzącej czas wykonania programu, wykonanie wykresu czasu wykonania programu w funkcji długości wejściowej listy o rozmiarach n= 10,50, 100, 500, 1000, 5000,10000 - 2 punkty