**Insertion Sort:**

Jest efektywny dla niewielkiej liczby elementów. Jego złożoność wynosi O(). Jest algorytmem stabilnym.

Najlepszym przypadkiem jest tablica danych wejściowych uporządkowana rosnąco. Wtedy w każdym przebiegu pierwszej pętli (pętla for) przy pierwszym sprawdzeniu warunków pętli while zachodzi arr[j]<key, czyli pętla while nie wykonuje się ani razu. Całkowity czas wykonywania algorytmu wyrażony jest funkcją liniową, zatem Insertion Sort działa w czasie O(n).

Najgorszym przypadkiem jest tablica danych wejściowych uporządkowana malejąco. W każdym przebiegu pętli for przy sprawdzaniu warunków pętli while zachodzi arr[j]>key, więc pętla while w każdym i-tym przebiegu pętli for jest wywoływana i-1 razy.

**Shell Sort:**

To algorytm działający w miejscu.Nie jest stabilny. Korzysta z porównań elementów. Na początku zajmuje się sortowanie elementów tablicy położonych daleko od siebie, a później stopniowo odstępy między sortowanymi elementami są zmniejszane.

Przypadkiem optymistycznym jest tablica danych wejściowych uporządkowanych, wtedy złożoność wynosi *O*(*n*1,14). Przypadkiem pesymistycznym jest jest tablica danych wejściowych uporządkowanych odwrotnie. Shell sort jest bezkonkurencyjny w klasie *O*(*n*2) algorytmów sortujących przy sortowaniu zbiorów nieuporządkowanych i zbiorów posortowanych odwrotnie, czyli w przypadku ogólnym.

**Selection Sort:**

Polega na wyszukaniu elementu minimalnego z tablicy spośród elementów od i do końca tablicy. Następnie wartość minimalna jest zamieniana z elementem na pozycji i. Operacja jest wykonywana dla wszystkich indeksów sortowanej tablicy. Złożoność wynosi *O*(*n*2). Algorytm jest niestabilny. Sortowanie odbywa się w miejscu. Najlepszym przypadkiem jest tablica uporządkowana rosnąco, a najgorszym - malejąco. W obu przypadkach złożoność wynosi *O*(*n*2).

**Heap Sort:**

Jest to niestabilny algorytm. Złożoność czasowa wynosi O(n\*logn). Jest to algorytm sortowania w miejscu. Najlepszym przypadkiem jest tablica uporządkowana rosnąco, a najgorszym - malejąco. W obu przypadkach złożoność wynosi *O*(*n* log*n*).

**Quick sort:**

W przypadku typowym algorytm ten jest najszybszym algorytmem sortującym z klasy złożoności obliczeniowej *O*(*n* log*n*) . Jest to algorytm sortujący w miejscu, ale nie jest stabilny. W najgorszym przypadku, czyli kiedy tablica danych wejściowych uporządkowana jest rosnąco lub malejąco, to złożoność wynosi *O*(*n*2). Najlepszym przypadkiem dla tego algorytmu jest sytuacja, w której pivot dzieli listę na dwie równe połowy, przechodząc dokładnie przez środek. Złożoność wynosi *O*(*n* log*n*).

WYKRESY:

W poniższych wykresach, tam gdzie rodzaj danych wejściowych nie był istotny, używane były tablice wejściowych danych losowych.





















