Algorytmy i struktury danych

Sprawozdanie z zadania w zespołach nr. 1
prowadząca: dr hab. inż. Małgorzata Sterna, prof PP

Algorytmy sortujące

autorzy:

Piotr Więtczak nr indeksu 132339,
Tomasz Chudziak nr indeksu 136691

22 marca 2018

1 Implementacja algorytmów sortujących

Do implementacji metod sortowania posłużyliśmy się językiem $C++$, każda metoda została napisana w odrębnej funkcji, która za parametry przyjmuje kolejno: wskaźnik na tablicę, rozmiar sortowanej tablicy oraz jako ostatni wartość opcjonalną "reverse" typu bool, która odpowiada za to czy tablica będzie posortowana malejąco. Do mierzenia czasu poszczególnych metod użyliśmy klasy *std* :: *chrono* :: *high* :: *resolution_clock* z biblioteki *chrono*.

2 Badana zależność czasu obliczeń $t[s]$ od liczby sortowanych elementów n.

W celu lepszego przedstawienia otrzymanych danych podzieliliśmy metody na dwie grupy, "wolne"(Insertion Sort, Selection Sort, Bubble Sort) i szybkie"(Counting Sort, Quick Sort, Merge Sort, Heap Sort).

2.1 Metody "wolne"

2.1.1 Opis algorytmów "wolnych"

Insert Sort

Zalety:

- działa w miejscu
- stabilny

Wady:

-

Inne cechy:

- zachowanie naturalne

Tabela przedstawiająca złożoność obliczeniową dla przypadków optymistycznego, średniego i pesymistycznego

	złożoność obliczeniowa dla przypadku optymistycznego	złożoność obliczeniowa dla przypadku średniego	złożoność obliczeniowa dla przypadku pesymistycznego
Insert Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$

Tablica 1: Tablica złożoności obliczeniowej dla metody Insert Sort

Selection Sort

Zalety:

- działa w miejscu
- stabilny

Wady:

-

Inne cechy:

- zachowanie naturalne

Tabela przedstawiająca złożoność obliczeniową dla przypadków optymistycznego, średniego i pesymistycznego

	złożoność obliczeniowa dla przypadku optymistycznego	złożoność obliczeniowa dla przypadku średniego	złożoność obliczeniowa dla przypadku pesymistycznego
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$

Tablica 2: Tablica złożoności obliczeniowej dla metody Selection Sort

Bubble Sort

Zalety:

- działa w miejscu
- stabilny

Wady:

-

Inne cechy:

- zachowanie naturalne

Tabela przedstawiająca złożoność obliczeniową dla przypadków optymistycznego, średniego i pesymistycznego

	złożoność obliczeniowa dla przypadku optymistycznego	złożoność obliczeniowa dla przypadku średniego	złożoność obliczeniowa dla przypadku pesymistycznego
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$

Tablica 3: Tablica złożoności obliczeniowej dla metody Bubble Sort

Tabela ilustrująca zależności czasu sortowania od ilości elementów dla metod "wolnych", zakres liczb $[1,n]$.

	Insertion Sort	Selection Sort	Bubble Sort

Tablica 4: Wyniki badań zależności czasu od iloci elementów dla metod "wolnych"

Wykres ilustrujący zależności czasu sortowania od ilości elementów dla metod "wolnych", zakres liczb $[1,n]$.

2.2 Metody "szybkie"

Quick Sort

Zalety:

- działa w miejscu
- stabilny

Wady:

-

Inne cechy:

- zachowanie nie naturalne
- korzysta z metody "dziel i rządź"

Tabela przedstawiająca złożoność obliczeniową dla przypadków optymistycznego, średniego i pesymistycznego

	złożoność obliczeniowa dla przypadku optymistycznego	złożoność obliczeniowa dla przypadku średniego	złożoność obliczeniowa dla przypadku pesymistycznego
Quick Sort	$O(n\log_2(n))$	$O(n\log_2(n))$	$O(n^2)$

Tablica 5: Tablica złożoności obliczeniowej dla metody Quick Sort

Merge Sort

Zalety:

- algorytm asymptotycznie optymalny

Wady:

- nie działa w miejscu
- wrażliwy na dane wejściowe

Inne cechy:

- korzysta z metody ”dziel i rządź”

Tabela przedstawiająca złożoność obliczeniową dla przypadków optymistycznego, średniego i pesymistycznego

	złożoność obliczeniowa dla przypadku optymistycznego	złożoność obliczeniowa dla przypadku średniego	złożoność obliczeniowa dla przypadku pesymistycznego
Merge Sort	$O(n\log_2(n))$	$O(n\log_2(n))$	$O(n^2)$

Tablica 6: Tablica złożoności obliczeniowej dla metody Merge Sort

Heap Sort

Zalety:

- działa w miejscu

Wady:

- wrażliwy na dane wejściowe

Inne cechy:

- zachowanie nienaturalne
- korzysta ze stogów

Tabela przedstawiająca złożoność obliczeniową dla przypadków optymistycznego, średniego i pesymistycznego

	złożoność obliczeniowa dla przypadku optymistycznego	złożoność obliczeniowa dla przypadku średniego	złożoność obliczeniowa dla przypadku pesymistycznego
Heap Sort	$O(n)$	*kek*	$O(n^2)$

Tablica 7: Tablica złożoności obliczeniowej dla metody Heap Sort

Counting Sort

Zalety:

-

Wady:

- nie działa w miejscu
- ograniczony ze względu na zakres sortowanych liczb
- mało wydajny dla danych z dużego przedziału

Inne cechy:

-

Tabela przedstawiająca złożoność obliczeniową dla przypadków optymistycznego, średniego i pesymistycznego

	złożoność obliczeniowa dla przypadku optymistycznego	złożoność obliczeniowa dla przypadku średniego	złożoność obliczeniowa dla przypadku pesymistycznego
Counting Sort	$O(n)$	$O(n)$	$O(n)$

Tablica 8: Tablica złożoności obliczeniowej dla metody Counting Sort

Tabela ilustrująca zależności czasu sortowania od ilości elementów dla metod "szybkich", zakres liczb $[1,n]$.

	Quick Sort	Merge Sort	Heap Sort	Counting Sort

Tablica 9: Wyniki badań zależności czasu od iloci elementów dla metod "szybkich"

Wykres ilustrujący zależności czasu sortowania od ilości elementów dla metod "szybkich", zakres liczb $[1,n]$.

Spis treści