Karol Kuchnio, s21912

```
range(0, 180000)
```

Algorithm	*Random order of numbers*	*Sorted Numbers*	*Reverse sorted numbers*
HeapSort	0.000000398	0.000000405	0.000000399
QuickSort	0.000000233	0.000000228	0.000000225
BubbleSort	0.000000275	0.00000027	0.000000269

```
range(0, 90000):
```

Algorithm	*Random order of numbers*	*Sorted Numbers*	*Reverse sorted numbers*
HeapSort	0.000000487	0.000000494	0.000000545
QuickSort	0.000000294	0.000000286	0.000000318
BubbleSort	0.000000318	0.000000307	0.000000311

```
range(0, 900):
```

Algorithm	*Random order of numbers*	*Sorted Numbers*	*Reverse sorted numbers*
HeapSort	0.000000885	0.00000113	0.000000861
QuickSort	0.000000514	0.000000497	0.000000505
BubbleSort	0.00000052	0.00000056	0.000000542

```
range(1):
```

Algorithm	*Random order of numbers*	*Sorted Numbers*	*Reverse sorted numbers*
HeapSort	0.0000013	0.0000011	0.000001
QuickSort	0.0000016	0.0000007	0.0000007
BubbleSort	0.0000024	0.0000007	0.0000007

Wynik mogą zawierac błędy pomiarowe wynikające z obciażenia maszyny na której był uruchamiany program porównujacy algorytmy.

Pomiary zostały wykonane na stu wywołaniach danej funkcji sortującej, tj. Bubble sort, Quick sort, Heap sort, następnie z w.w podane wyniki są srednia wartoscią z tego pomiaru.

Wnioski

Powyższe pomiary nie wygladają najlepiej w przypadku tablic mieszanych, gdzie tak naprawde najlepiej wydajnosciowo powinien wypasc HeapSort, natomiast w 1,2,3 tablicy, gdzie mamy najwieksze zbiory liczb QuickSort ma najlepsza wydajność, dopiero w tablicy 4 widzimy ze zamienia się pozycja z HeapSort.

W przypadku posortowanych talic widzimy, ze we wszystkich tablicach QuickSort(ostatnia tablica egzekwo z BubbleSort) radzi sobie najlepiej z sortowaniem.

Natomiast w tablicy odwrotnie posortowanej QuickSort znowy wypada najlepiej, jedynie w drugiej tabeli widzimy, ze minimalnie przegrywa z BubbleSort.

Możliwe, ze ilosc iteracji podczas przeprowadzonych sortowan wplyneła na powyzsze wyniki. Temat powinien zostac zgłebiony oraz przeprowdzony na maszynie wolnej od dodatkowego obciazenia. Według informacji zanezionych w literaturze generalnie HeapSort jest szybszy od Bubble sort, natomiast sa tutaj warunki, w których BubbleSort może radzic sobie lepiej:

- Jeżeli tablica jest posortowana wtedy BubbleSort wygrywa swoja zloznoscia O(n), gdzie HeapSort jest o zlozonosci: s O(n log n)

- Implementacja danego algorytmu, tworzenie sterty może wiazac się z ‘przerostem formy nad trescią’, gdzie w przypadku malych danych wejsciowych, BubbleSort może być lepszym rozwiazaniem.