1 Lista4 Zadanie5

1.1 Algorytm

Jako rozwiązanie tego zadania skorzystam z algorytmu sortowania, mianowicie z MergeSort.
Aby wyznaczyć liczbę inwersji dzięki MergeSort, wystarczy mała modyfikacja metody Merge().
Aby pokazać, różnice, zademonstuję implementacje zwykłej metody merge oraz zmodyfikowanej metody merge.
W przypadku scalania dwóch części tablicy A, mamy operatory i i j które operują odpowiednio na lewej i prawej części tablicy. Tak więc w momencie scalania:
Jeśli $l_A[i] > p_A[j]$ to dodajemy liczbę inwersji

```
if l_A[i]>p_j[j]:
    inversions += len(l_a) - i
    #... dalsze instrukcje
```

Instrukcja len(l_a) - i definiuje nam ile elementów mniejszych od $p_j[j]$ przeskoczyliśmy, wstawiając to na odpowiednie miejsce.
W wyniku takiej zmiany otrzymujemy liczbę inwersji.

1.2 Złożoność

Za każdym krokiem dzielimy podbroblem na podproblemy dwa razy krótsze. Więc rekurencje występującą w MergeSort można zapisać:

$$T(n) = T(\frac{n}{2}) + f(n)$$

f(n) to koszt operacji merge, który jest liniowy. Więc mamy, że $f(n) = \Theta(n)$
Więc na mocy twierdzenia o rekursji uniwersalnej:

$$T(n) = O(nlogn)$$