Algorytmy i Struktury Danych - lista 3

1. Napisz funkcję `void rotate(T *p, T *s, T *k)` zamieniającą miejscami obszary `[p,s)` i `[s,k)` o długościach m i n.

   (a) Pokaż, że można zrobić to przy użyciu trzech inwersji i wówczas algorytm sprowadza się do wykonania $m+n$ zamian czyli $3(m+n)$ podstawień.

   (b) Dla przypadku, gdy $m = n$ napisz procedurę wykonującą tylko m zamian czyli $3m$ podstawień.

   (c) Dla przypadku, gdy $m = n+1$ napisz procedurę wykonującą tylko $2m$ podstawień.

2. Napisz funkcję `void in_place_merge(T *p, T *s, T *k)`, która nie używa dodatkowej pamięci. Wsk. znajdź takie $p1 \in [p,s)$ i $k1 \in [s,k)$ aby po wykonaniu `rotate(p1,s,k1)` element $*s$ stanowił medianę i skorzystaj z rekurencji.

3. Udowodnij, że złożoność `in_place_merge` wynosi $O(n \log n)$, a złożoność `in_place_merge_sort`, który jej używa, wynosi $O(n \log^2 n)$.

4. Pokaż, że `selection_sort` nie jest stabilny. Ile dodatkowej pamięci potrzeba, by uczynić go stabilnym ?

5. Napisz klasę `priority_queue` opartą na kopcu zrealizowanym w tablicy. (Dziećmi elementu `t[i]` są `t[2*i+1]` oraz `t[2*i+2]`. Ojcem `t[i]` jest `t[(i-1)/2]`.) Zaimplemetuj metody `put(T)`, `T max()`, `T get_max()` oraz `bool is_empty()`.

6. Napisz program nie zawierający instrukcji `if` ani `switch`, który policzy ile razy występuje każdy znak ASCII w pliku podanym jako argument programu.