**Анализ на задача**

**Конгруентност**

Съотношенията между буквите могат да се запазят, ако се използват последователни букви. Най-малката по азбучен ред ще се получи, когато заменим най-малката в дадената дума с A.

Това разсъждение води до следния алгоритъм за намиране на най-малката дума:

1. Намираме различните букви в дадената дума. Означаваме броя им с *n*.
2. Ако *n*=26, няма друга дума, конгруентна на дадената: тя е единствената, т.е. и най-малката, и най-голямата – извеждане входната дума два пъти.
3. Иначе правим списък *L* на срещаните букви.
4. Подреждаме списъка растящо по големина (т.е., в азбучен ред).
5. На всяка буква от подредения списък *L* съпоставяме последователно букви от азбуката, започвайки от A.
6. Заменяме в дадената дума всички букви със съответните им, като получаваме най-малката по азбучен ред дума, конгруентна на дадената.
7. Можем да приложим подобна стратегия и за най-голямата дума, като започнем съпоставянето в подредения списък *L* в обратен ред и съпоставяме букви намаляващо, започвайки от Z. Това обаче може да се избегне, като вземем предвид, че всяка буква от най-голямата дума ще е константно изместена от съответната ѝ в най-малката дума.

Добрата реализация на този алгоритъм позволява да не използваме два големи буфера: за съхраняване на входа и за преработката му – всичко може да стане само с един буфер.

Ако означим броя на буквите в думата с *N*, сложността на този алгоритъм в най-лошия случай е O(28*N*): едно сканиране за въвеждане и статистика, до 25 за промяна на буквите и две за извеждане. Можем да заменим 25-те сканирания с двоично търсене на съответната буква в подредения списък, с което коефициентът за тях да падне на 5 (в най-лошия случай), давайки обща сложност O(8*N*). Ако ползваме проста динамична идея за съответстващата буква, общата сложност пада до O(4*N*).

*Автор: Павлин Пеев*