# Longest Common Substring

Идея: Андрей Станкевич
Разработка: Геннадий Короткевич

Для решения этой задачи нужно придумать, как сжать информацию о каждой строке так, чтобы можно было легко проверить, что некоторая строка $w$ действительно является наибольшей общей подстрокой $s$ и $t$.

Оказывается, такое сжатие несложно выполнить. Рассмотрим все возможные $2^{k+1}$ битовых строк длины $k+1$ и выпишем битовую маску $b(s)$ из $2^{k+1}$ бит, обозначающую, какие из них являются подстроками $s$. Аналогично определим $b(t)$. Тогда, если у $b(s)$ и $b(t)$ есть общие биты, то у $s$ и $t$ есть общая подстрока длины $k+1$, а значит, строка $w$ длины $k$ не может являться их наибольшей общей подстрокой. В противном случае достаточно проверить, что $w$ входит в $s$ и в $t$ — а это верно, если хотя бы одна из строк $w\texttt{0}$, $w\texttt{1}$, $\texttt{0}w$ или $\texttt{1}w$ (длины $k+1$) входит в $b(s)$, и то же с $b(t)$.

Теперь воспользуемся методом динамического программирования. Пусть $f(n,p,b)$ — число строк длины $n$, последние $k$ бит которых равны $p$, а маска встреченных подстрок длины $k+1$ равна $b$. Для переходов “вперёд” переберём очередной $n+1$-й бит (назовём его $d$), допишем $d$ к строке $p$ справа, добавим подстроку $p$ в маску $b$, и отрежем левый бит от $p$.

Состояний и переходов у данной функции $O(n \cdot 2^k \cdot 2^{2^{k+1}})$, что не слишком много в заданных ограничениях.

В последней части решения — переборе масок $b(s)$ и $b(t)$ — можно перебирать только пары масок, не имеющих общих единичных битов. Таких пар ровно $3^{2^{k+1}}$, что тоже не слишком много.

Приведённый алгоритм работает только в случаях $n > k$ и $m > k$, поэтому случаи $n = k$ или $m = k$ надо рассмотреть отдельно.