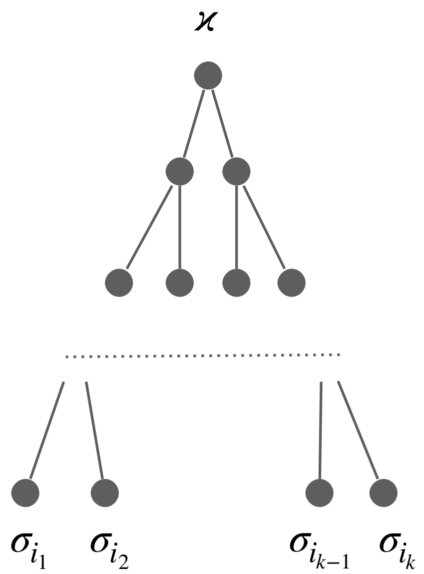
Алгоритъм на Lempel-Ziv. Построяване на суфиксни масиви.
08.01.2021 г.

Една от мотивациите да може да строим суфиксно дърво on-line е, че това позволява компресия, при това ефективна такава (линейно време и теоритично смачкване на данните, - възможно най-много - алгоритъма на Lempel-Ziv).

Връзка между суфиксни масиви, суфиксни дървета и компресия.

Нека имаме текст $T = t_1 t_2 \dots t_n$, $t_i \in \Sigma$, $i = \overline{1,n}$. Искаме да представим този текст T с възможно най-малко битове (най-малко памет). В тази връзка, когато Шенън е започнал да разсъждава върху този въпрос през 60-те години на миналия век, той заключил следното: ако искаме да представим символите независимо един от друг, т.е. искаме някаква компресия $\underset{kappa}{\varkappa} : \Sigma \to \{0,1\}^*$, така че ако имаме две букви $\sigma_1, \sigma_2 \in \Sigma$ и $\sigma_1 \neq \sigma_2$, $\underbrace{\varkappa(\sigma_1) \not\prec_{pref} \varkappa(\sigma_2)}_{(*)}$. Идеята е следната: ако имаме такава функция $\varkappa(\sigma)$, то тогава може да компресираме текста $T \mapsto \varkappa(t_1)\varkappa(t_2)\dots\varkappa(t_n)$. Условието $(*)$ означава, че няма многозначности. Може да си представим $\varkappa$ като едно двоично дърво с нули и единици:



и някъде по листата на това дърво седи някаква пермутация на символите σ_i. Разсъждавайки по този начин стигаме до следната парадигма - искаме да минимизираме $\sum_{i=1}^{n} |\varkappa(t_i)|$, т.е. $\min \sum_{i=1}^{n} |\varkappa(t_i)| = M$ което означава, че като преброим колко пъти се среща всеки символ в текста,
$M = \min \sum_{\sigma \in \Sigma} \underbrace{\mathrm{occ}_T(\sigma)}_{\text{брой срещания на } \sigma} \underbrace{|\varkappa(\sigma)|}_{\text{дължина на } \sigma}$. Оптимумът се достига при

$|\varkappa(\sigma)| = -\log_{(2)} \frac{\mathrm{occ}_T(\sigma)}{|T|}$ - честота на символа σ в текста T. Крайният резултат казва, че честосрещаните символи в текстамискаме да ги компресираме с възможно най-малко битове, така че те да окажат най-малко влияние върху цялата сума. И обратното - рядко срещаните символи съответно ще бъдат кодирани с по дълго представяне. Това е мясгтото, от където се появява тази ентропия, за която става въпрос през цялото време.

Ако означим $p(\sigma) = \mathbb{P}(\sigma) = \frac{\mathrm{occ}_T(\sigma)}{|T|}$ да е честотата на срещане на символа σ в T, то $H(T) \stackrel{def.}{=} -\sum_{\sigma} p(\sigma) \ln p(\sigma)$ е ентропията на това колко добре може да компресираме един текст. Алгоритъм на Хъфман (Huffman). Това е най-простата схема, ако не искаме да вземаме под внимание контекста, в който символите се срещат, с която човек може да реши проблема за компресията. Обаче обикновено има зависимост между символите и ние бихме искали да отчетем тази зависимост. Алгоритъма на Lempel-Ziv постига точно това.