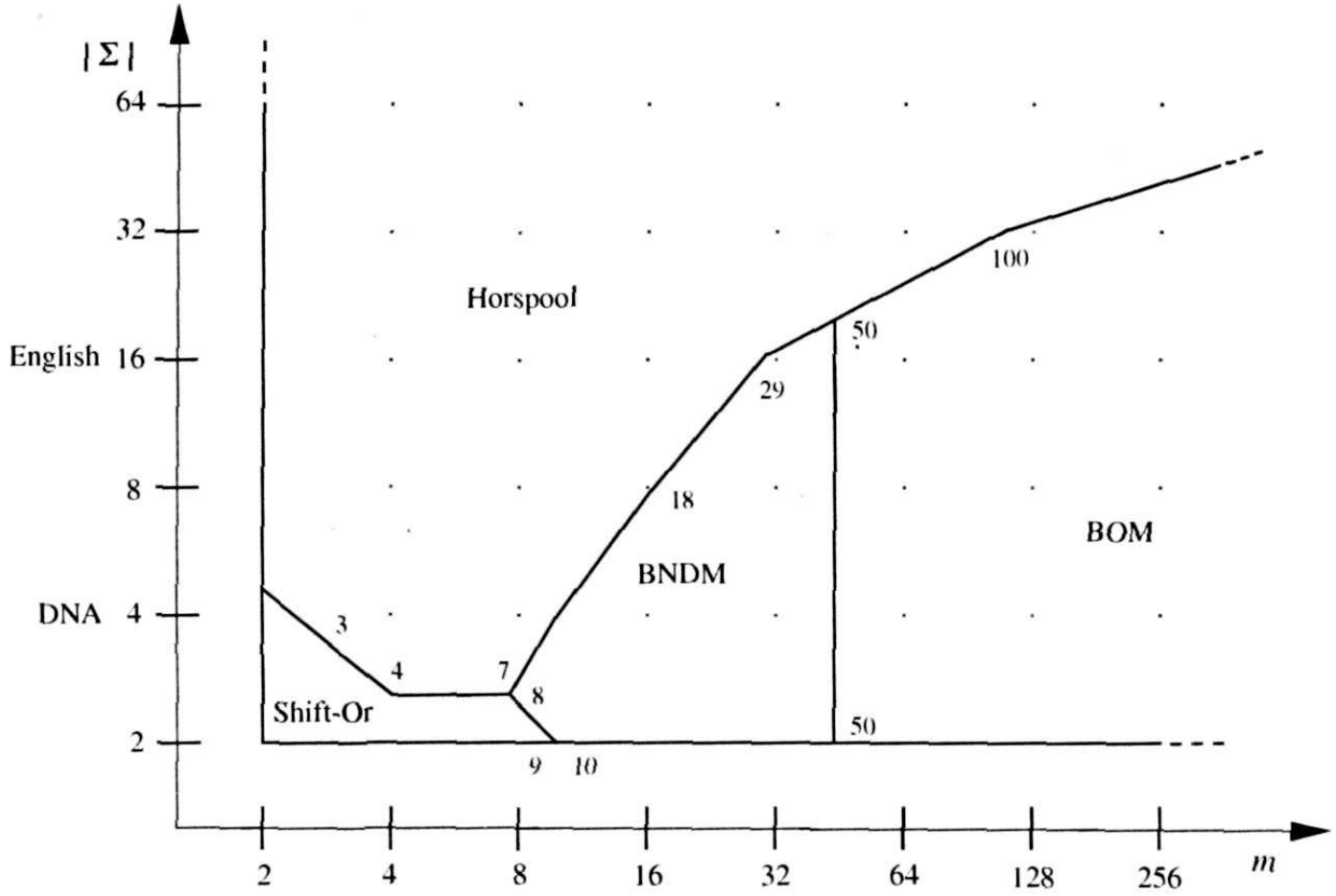
Overview of string matching algorithms

Algorithm	Worst case	Average case	Comp.	Note
Trivial	$O(nm)$	$O(n+m)$	Y	simple
DFA	$O(n+\sigma m)$	$O(n+\sigma m)$	N	real-time
(K)MP	$O(n+m)$	$O(n+m)$	Y	
Boyer Moore	$O(nm), O(n+m)$	$O(\frac{n}{\sigma}+m)$	N	
Rabin-Karp	$O(nm)$	$O(n+mk)$	N	randomized
Shift-and	$O(n+m+\sigma)$	$O(n+m+\sigma)$	N	m-bit register
BNDM	$O(nm+\sigma)$	$O(n\frac{\log_\sigma m}{m}+m+\sigma)$	N	m-bit register

Multiple patterns: Aho Corasick $O((n+m)\log\sigma+k)$

2D patterns: Baker Bird $O((n+m)\log\sigma)$

Fastest algorithm for various m and σ



Navarro, Raffinot (2002) Flexible Pattern Matching in Strings.

Random texts, 32-bit numbers

Nepovinná domáca úloha

V Boyer-Moorovom algoritme s použitím iba slabého pravidla dobrého sufixu nájdite príklady P a T, ktoré vyžadujú kvadratický počet porovnaní, aj keď sa P v T nenachádza.

Boyerov-Moorov algoritmus 1977

```
1  i=0;
2  while(i<=n-m){
3     j=m-1;
4     while(j>=0 && P[j]==T[i+j]){
5        j--;
6     }
7     if(j<0){ print i; }
8     i += skip;
9  }
```

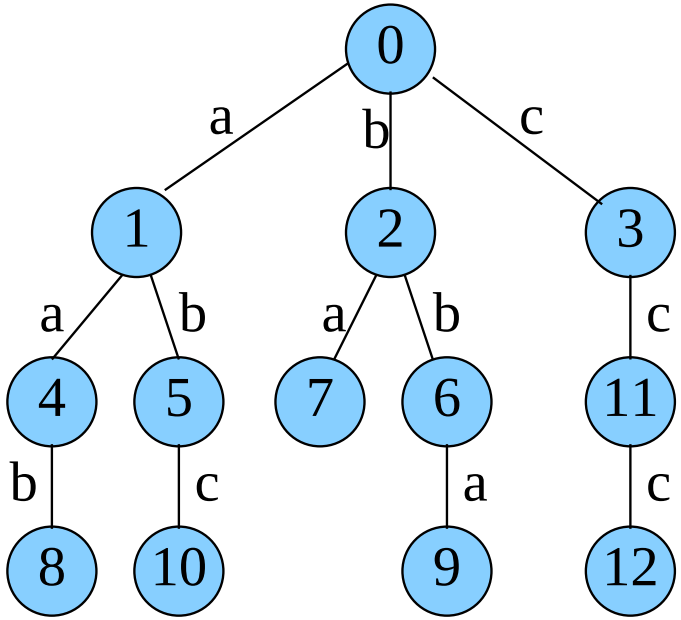
Good suffix rule: skip $\gamma[j]$:

$\gamma[j] = m - \max\{k \mid 0 \leq k < m \wedge P[j+1..m-1] \sim P[0..k-1]\}$

$P \sim Q$ iff P is a suffix Q or Q is a suffix of P

LZW compression (Lempel–Ziv–Welch 1984)

Used in GIF, popular UNIX compression in the past, formerly patent issues



Original text: aabbaabbabcccccc

Compressed: 1, 1, 2, 2, 4, 6, 5, 3, 11, 12