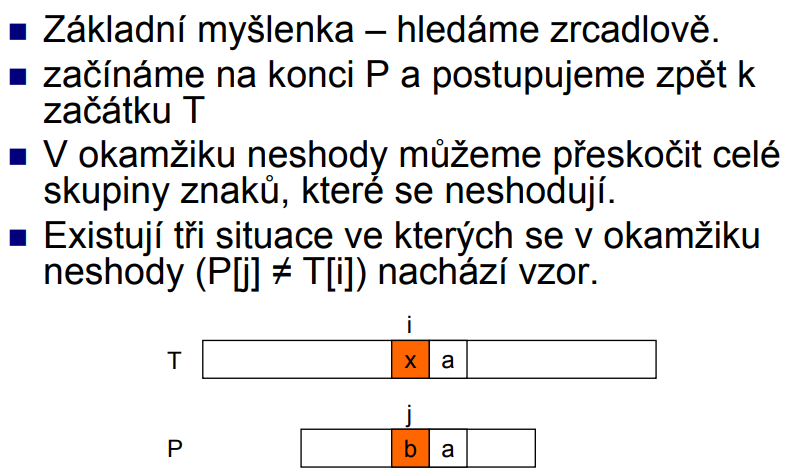
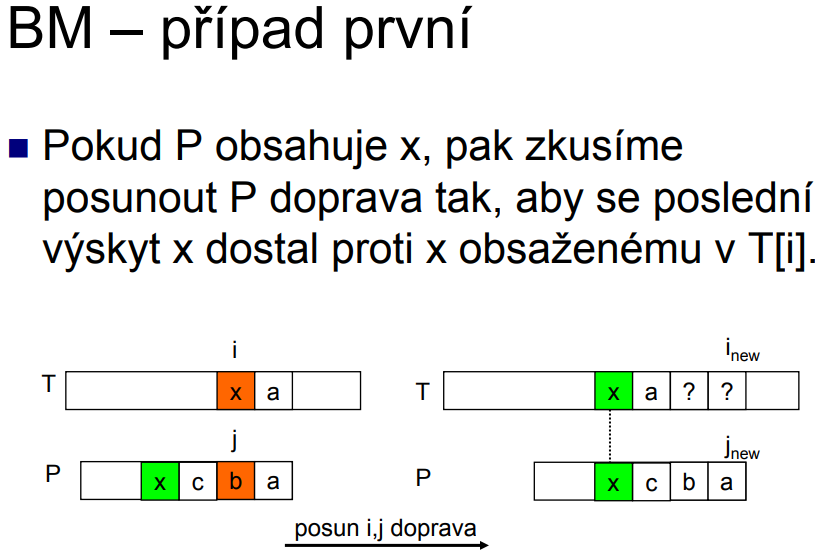
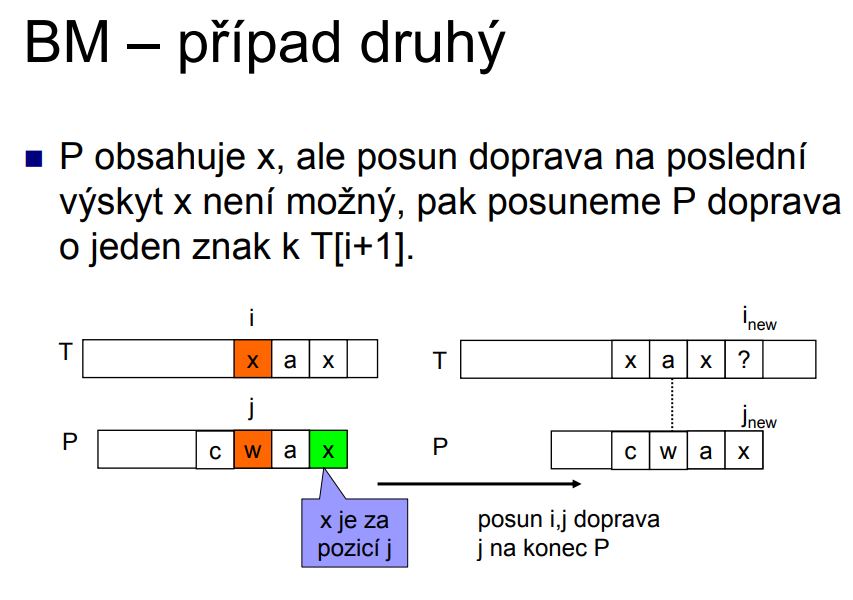
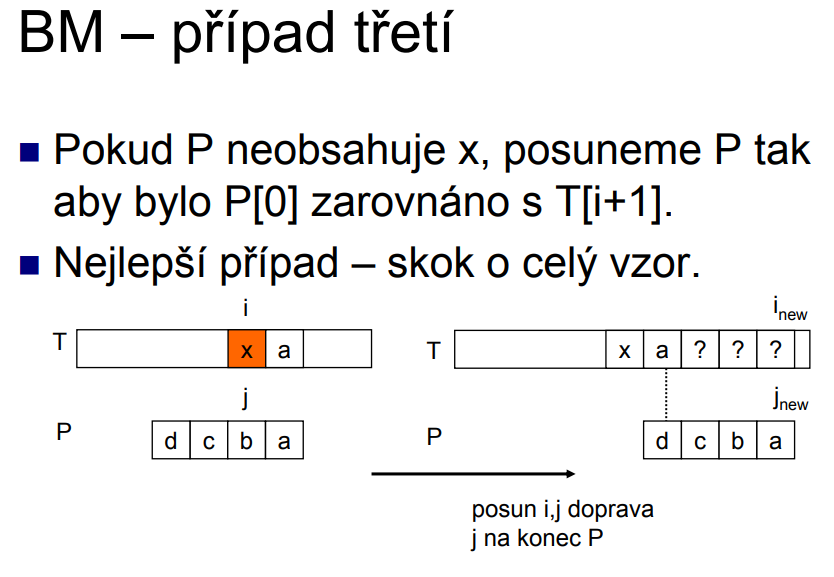
27. Prohledávání řetězců - Boyer-Moore, Rabin-Karp - princip, srovnání

# Boyer-Moore



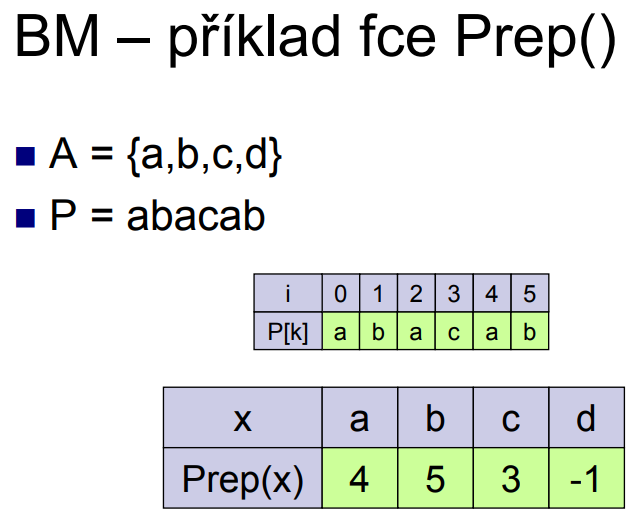






# BM – preprocesing

* Stejně jako u KMP i zde určujeme posuny předem analýzou vzoru
* Pro preprocesing používáme zobrazení všech znaků použité abecedy A do množiny celých čísel
* Funkce Prep()
* Pro libovolný znak x z A volíme Prep(x) jako:
  + Největší index i pro který platí, že P[i] == x, nebo
  + -1 pokud žádný takový index v P neexistuje



# Zhodnocení

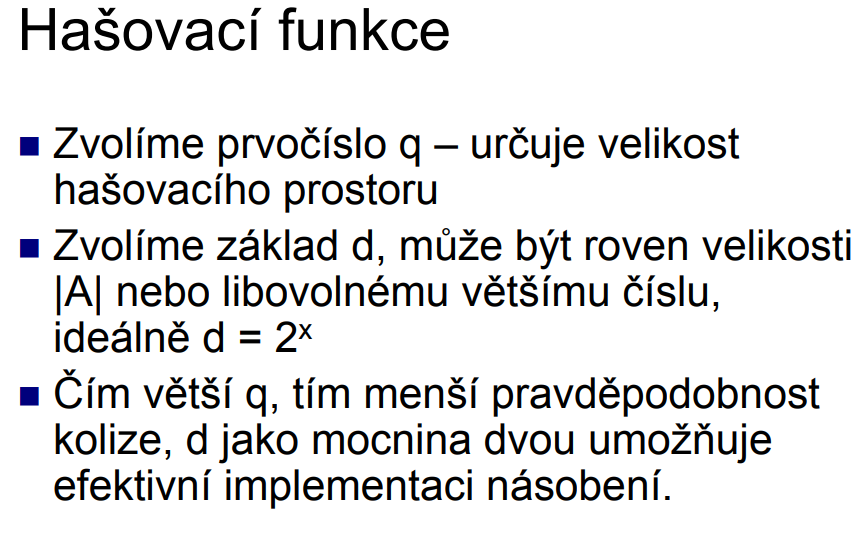
* Rychlý pro velkou abecedu, pomalý pro malou (podobně jako přirozené vyhledávání)
* Pro stejné abecedy bude ale BM rychlejší než přirozené

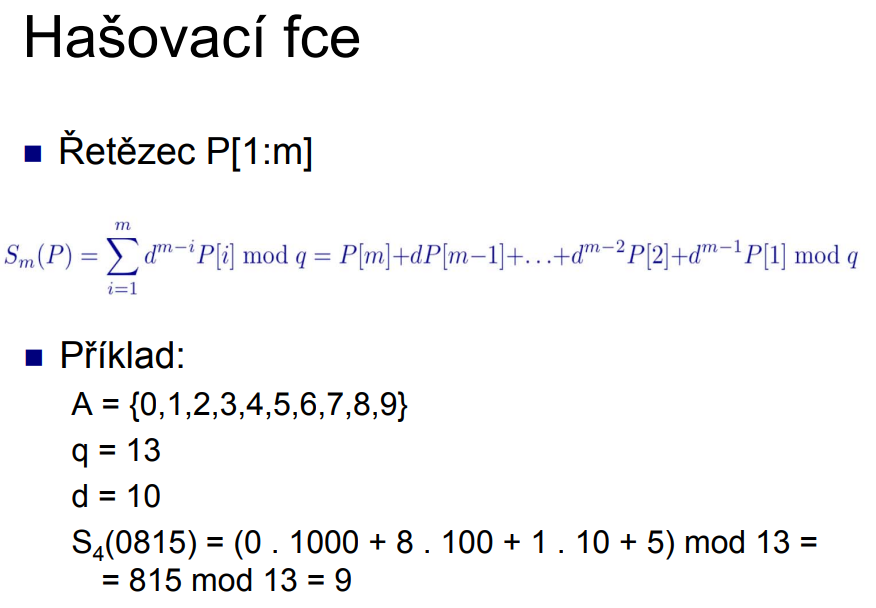
# Rabin-Karp algoritmus

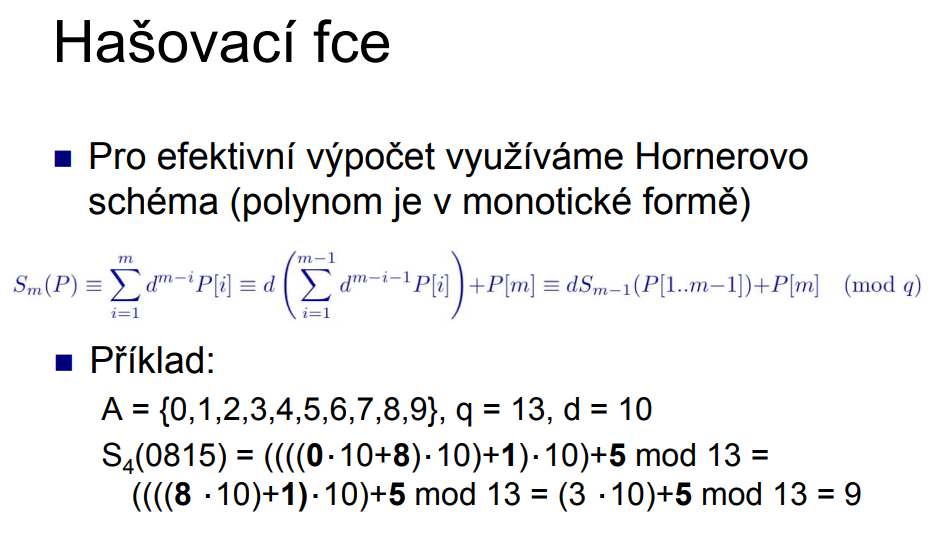
* Založen na použití hašování
* Vypočteme hash pro vzor P (délky M) a pro každý podřetězec řetězce T délky M
* Procházíme řetězcem T ale místo jednotlivých znaků porovnáváme hash každého podřetězce a vzoru
* V případě shody provedeme test podřetězce a vzor znaku po znaku – ochrana pro kolizi hashe

# Jakou zvolit hashovací funkci?

* Klíčová volby pro efektivitu celého algoritmu
* Podřetězec se posouvá po znaku => části podřetězců jsou shodné
* Potřebujeme funkci, která umožní vypočítat hash následujícího podřetězce s využitím již vypočtených hashů







# Rabin-Karp zhodnocení

* Přes stejnou složitost běží algoritmus pomaleji než KMP na stejných datech (díky režii výpočtů)
* Algoritmus lze ovšem snadno rozšířit na vyhledávání více slov (vzorů) najednou
* Algoritmus se efektivně uplatňuje například v detekci plagiátů