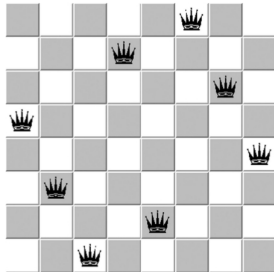CH-231-A

Algorithms and Data Structures

ADS

Lecture 39

Dr. Kinga Lipskoch

Spring 2020

The Eight-Queens Problem

▶ The eight queens problem is a classical puzzle of positioning eight queens on an 8×8 chessboard such that no two queens threaten each other.

▶ This a classical textbook backtracking problem.

Eight Queens: Representation

- ▶ What is concise, efficient representation for an n-queens solution, and how big must it be?
- ▶ Since no two queens can occupy the same column, we know that the n columns of a complete solution must form a permutation of n.
- ▶ By avoiding repetitive elements, we reduce our search space to just $8! = 40, 320$ quick for any reasonably fast machine.
- ▶ The critical routine is the candidate constructor.
- ▶ We repeatedly check whether the kth square on the given row is threatened by any previously positioned queen.
- ▶ If so, we move on, but if not we include it as a possible candidate.
- ▶ Algorithm can find the $365, 596$ solutions for $n = 14$ in minutes.

String Matching

- ▶ Text-editing programs often need to find all occurrences of a pattern in the text.
- ▶ Among their many other applications, string-matching algorithms search for particular patterns in DNA sequences.
- ▶ Internet search engines also use them to find web pages relevant to queries.

String Matching Problem (1)

- ▶ We assume that the text is an array $T[1..n]$ of length n and that the pattern is an array $P[1..m]$ of length $m \leq n$.
- ▶ We further assume that the elements of P and T are characters drawn from a finite alphabet Σ.
- ▶ For example, we may have $\Sigma = \{0, 1\}$ or $\Sigma = \{a, b, ..., z\}$.
- ▶ The character arrays P and T are often called strings of characters.

String Matching Problem (2)

▶ The pattern P occurs with shift s in text T (or, P occurs beginning at position $s + 1$ in text T)

▶ If P occurs with shift s in T, then we call s a valid shift; otherwise, we call s an invalid shift.

▶ The string-matching problem is the problem of finding all valid shifts with which a given pattern P occurs in a given text T.
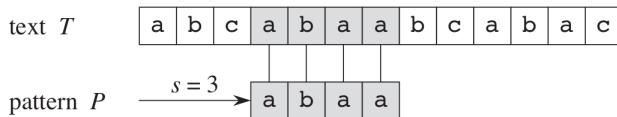
Notation and Terminology

- Σ^* the set of all finite-length strings formed using characters from the alphabet Σ
- ϵ is the zero-length empty string
- $|x|$ is the length of a string x
- xy is the concatenation of two strings x and y
- $w \sqsubset x$ means w is a prefix of a string x, i.e., $x = wy$
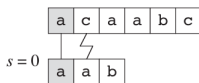- $w \sqsupset x$ means w is a suffix of a string x, i.e., $x = yw$

Naive String-Matching Algorithm

The naive algorithm finds all valid shifts using a loop that checks the condition $P[1..m] = T[s+1..s+m]$ for each of the $n - m + 1$ possible values of s.
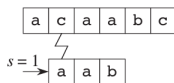
NAIVE-STRING-MATCHER(T, P)

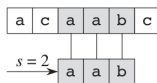1 $n = T.length$
2 $m = P.length$
3 **for** $s = 0$ **to** $n - m$
4   **if** $P[1..m] == T[s+1..s+m]$
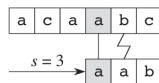5   print "Pattern occurs with shift" s

(a)   (b)   (c)   (d)

Naive String-Matching Time Complexity

- ▶ For example, consider the text string a^n (a string of n a's) and the pattern a^m.
- ▶ For each of the $n - m + 1$ possible values of the shift s, the implicit loop on line 4 to compare corresponding characters must execute m times to validate the shift.
- ▶ The worst-case running time is thus $\Theta((n - m + 1)m)$, which is $\Theta(n^2)$ if $m = \lfloor n/2 \rfloor$.
- ▶ The naive string-matcher is inefficient because it entirely ignores information gained about the text for one value of s when it considers other values of s.

String Matching with Finite Automata

- ▶ Many string-matching algorithms build a finite automaton that scans the text string T for all occurrences of the pattern P.
- ▶ The matching time used after preprocessing the pattern to build the automaton is $\Theta(n)$.
- ▶ The time to build the automaton, however, can be large if Σ is large.
- ▶ The Knuth-Morris-Pratt algorithm has a clever way around this problem.

Knuth-Morris-Pratt Algorithm

- ▶ It is a linear-time string-matching algorithm due to Knuth, Morris, and Pratt.
- ▶ Its matching time is $\Theta(n)$ using just an auxiliary function π, which we precompute from the pattern in time $\Theta(m)$ and store in an array $\pi[1..m]$.

Prefix Function for a Pattern (1)

(a)

(b)

(c)
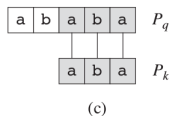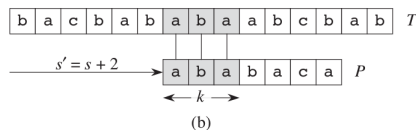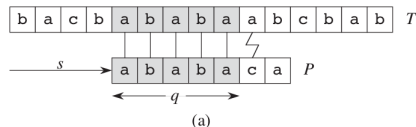
Prefix Function for a Pattern (2)

- ▶ Subfigure (a) shows a particular shift s of a template containing the pattern $P = ababaca$ against a text T.
- ▶ For this example, $q = 5$ of the characters have matched successfully, but the 6th pattern character fails to match the corresponding text character.
- ▶ The information that q characters have matched successfully determines the corresponding text characters.
- ▶ Knowing these q text characters allows us to determine immediately that certain shifts are invalid.
- ▶ The shift $s' = s + 2$ shown in subfigure (b) of the figure, however, aligns the first three pattern characters with three text characters that must necessarily match.