String Matching

Inge Li Gørtz

String Matching

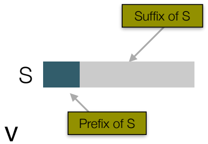
- String matching problem:
  - string T (text) and string P (pattern) over an alphabet Σ.
  - |T| = n, |P| = m.
  - Report all starting positions of occurrences of P in T.

P = a b a b a c a

T = b a c b a b a b a b a b a c a b

Strings

- ε: empty string
- prefix/suffix: v=xy:
  - x *prefix* of v, if y ≠ ε x is a *proper prefix* of v
  - y *suffix* of v, if y ≠ ε x is a *proper suffix* of v.



- Example: S = aabca
  - The suffixes of S are: aabca, abca, bca, ca and a.
  - The strings abca, bca, ca and a are proper suffixes of S.

String Matching

- Knuth-Morris-Pratt (KMP)
- Finite automaton

A naive string matching algorithm

| b | a | c | b | a | b | a | b | a | b | a | b | a | c | a | b |

a b a b a c a
a b a b a c a
a b a b a c a
a b a b a c a
a b a b a c a
a b a b a c a
a b a b a c a
a b a b a c a
a b a b a c a
a b a b a c a

Improving the naive algorithm

P = a a a b a b a

T = a a a b a a a b a b a b a c a b b

a a a b a b a

Improving the naive algorithm

P = a a a b a b a

T = a a a b a [redacted]

a a a b a b a

a a a a a b b a b a

Improving the naive algorithm

P = a a a b a b a

T = a a a b a a a [redacted]

a a a b a b a

a a a b a b a

a a a b a b a

Improving the naive algorithm

P = a a a b a b a

T = a a a b a a a a b a b a [redacted]

a a a b a b a

a a a b a b a

a a a b a b a

a a a a a a a b a b a

Improving the naive algorithm

P = a a a b a b a

T = a a a b a a a a b a b a [redacted]

a a a b a b a

a a a b a b a

a a a b a b a

a a a b a b a

If we matched 5 characters from T and then fail: compare failed character to 2nd character in P

If we matched 3 characters from T and then fail: compare failed character to 3nd character in P

If we matched all characters from T: compare next character to 2nd character in P

Improving the naive algorithm

P = a a a b a b a

matched		a	a	a	b	a	b	a
#matched	0	1	2	3	4	5	6	7
if fail compare to				3		2		2

If we matched 5 characters from T and then fail: compare failed character to 2nd character in P

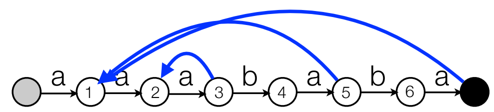
If we matched 3 characters from T and then fail: compare failed character to 3nd character in P

If we matched all characters from T: compare next character to 2nd character in P

Improving the naive algorithm

P = a a a b a b a

matched		a	a	a	b	a	b	a
#matched	0	1	2	3	4	5	6	7
if fail compare to				3		2		2



If we matched 5 characters from T and then fail: compare failed character to 2nd character in P

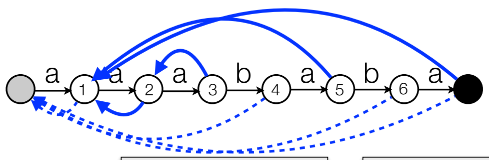
If we matched 3 characters from T and then fail: compare failed character to 3nd character in P

If we matched all characters from T: compare next character to 2nd character in P

Improving the naive algorithm

P = a a a b a b a

matched		a	a	a	b	a	b	a
#matched	0	1	2	3	4	5	6	7
if fail compare to	1	1	2	3	1	2	1	2



If we matched 5 characters from T and then fail: compare failed character to 2nd character in P

If we matched 3 characters from T and then fail: compare failed character to 3nd character in P

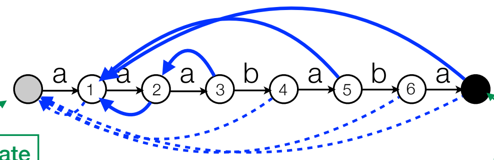
If we matched all characters from T: compare next character to 2nd character in P

KMP and π-array

- KMP: P = aaababa.

π-array

matched		a	a	a	b	a	b	a
#matched	0	1	2	3	4	5	6	7
if fail *go to*	0	0	1	2	0	1	0	1



starting state

accepting state

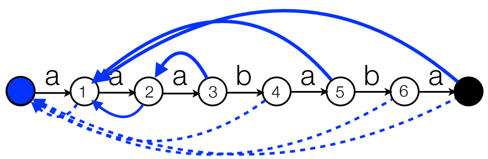
In state i after reading character j of T: P[1…i] is the longest prefix of P that is a suffix of T[1..j]

KMP and π-array

- KMP: P = aaababa.

π-array

matched		a	a	a	b	a	b	a
#matched	0	1	2	3	4	5	6	7
if fail *go to*	0	0	1	2	0	1	0	1

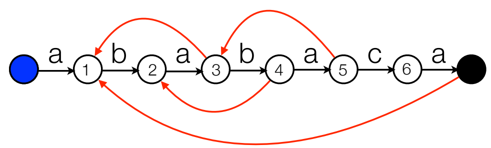


- Matching:

T = a a a b a a a b a b a a

KMP

- KMP: Can be seen as finite automaton with *failure links*:
  - Failure link: longest prefix of P that is a proper suffix of what we have *matched* until now.
  - In state i after reading T[j]: P[1..i] is the longest prefix of P that is a suffix of T[1…j].
  - Can follow several failure links when matching one character:



T = a b a b a a

KMP Analysis

- Analysis. |T| = n, |P| = m.
  - How many times can we follow a forward edge?
  - How many backward edges can we follow (compare to forward edges)?
  - Total number of edges we follow?
  - What else do we use time for?

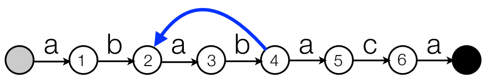
KMP Analysis

- Lemma. The running time of KMP matching is O(n).
  - Each time we follow a forward edge we read a new character of T.
  - #backward edges followed ≤ #forward edges followed ≤ n.
  - If in the start state and the character read in T does not match the forward edge, we stay there.
  - Total time = #non-matched characters in start state + #forward edges followed + #backward edges followed ≤ 2n.

Computation of failure links

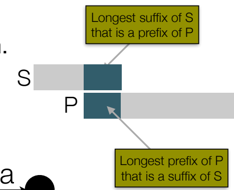
- Failure link: longest prefix of P that is a proper suffix of what we have *matched* until now.
- Computing failure links: Use KMP matching algorithm.

longest prefix of P that is a proper suffix of ‘abab'

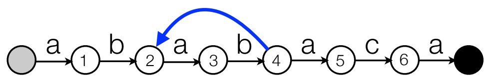


Computation of failure links

- Failure link: longest prefix of P that is a proper suffix of what we have *matched* until now.
- Computing failure links: Use KMP matching algorithm.

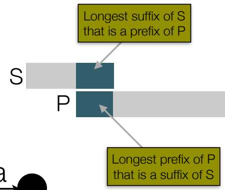


longest prefix of P that is a suffix of ‘bab'

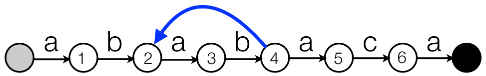


Computation of failure links

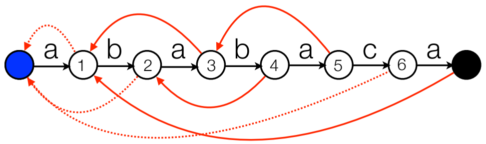
- Failure link: longest prefix of P that is a proper suffix of what we have *matched* until now.
- Computing failure links: Use KMP matching algorithm.



longest prefix of P that is a suffix of 'bab'

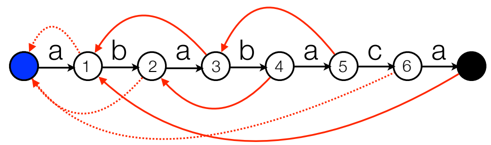


Can be found by using KMP to match 'bab'



Computation of failure links

- Computing failure links: As KMP matching algorithm (only need failure links that are already computed).
- Failure link: longest prefix of P that is a proper suffix of what we have *matched* until now.



	1	2	3	4	5	6	7
P =	a	b	a	b	a	c	a

Rabin-Karp

Fingerprinting

Rabin-Karp

- Fingerprint: construct randomized fingerprint for P and each substring of T of length m.
- Assume (wlog.) binary alphabet.

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P	1	0	1

T	1	0	1	0	1	1	0	1	0

F(P) = $2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

Rabin-Karp

- Fingerprint: construct randomized fingerprint for P and each substring of T of length m.
- Assume (wlog.) binary alphabet.

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P: | 1 | 0 | 1 |

T: | **1** | **0** | **1** | 0 | 1 | 1 | 0 | 1 | 0 |

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) =$

Rabin-Karp

- Fingerprint: construct randomized fingerprint for P and each substring of T of length m.
- Assume (wlog.) binary alphabet.

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P: | 1 | 0 | 1 |

T: | **1** | **0** | **1** | 0 | 1 | 1 | 0 | 1 | 0 |

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P: | 1 | 0 | 1 |

T: | **1** | **0** | **1** | 0 | 1 | 1 | 0 | 1 | 0 |

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P: | 1 | 0 | 1 |

T: | 1 | **0** | **1** | **0** | 1 | 1 | 0 | 1 | 0 |

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_2) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 = 2$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P

1	0	1

T

1	0	1	0	1	1	0	1	0

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_2) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 = 2$

$F(T_3) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P

1	0	1

T

1	0	1	0	1	1	0	1	0

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_2) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 = 2$

$F(T_3) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_4) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 1 = 3$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P

1	0	1

T

1	0	1	0	1	1	0	1	0

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_2) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 = 2$

$F(T_3) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_4) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 1 = 3$

$F(T_5) = 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 0 = 6$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P

1	0	1

T

1	0	1	0	1	1	0	1	0

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

P occurs in T at position s
$\Leftrightarrow$
$F(P) = F(T_s)$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_2) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 = 2$

$F(T_3) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_4) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 1 = 3$

$F(T_5) = 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 0 = 6$

$F(T_6) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_7) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 = 2$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P			
P	1	0	1

T									
T	1	0	1	0	1	1	0	1	0

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

P occurs in T at position s
⇔
$F(P) = F(T_s)$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = \mathbf{5}$

$F(T_2) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 = 2$

$F(T_3) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = \mathbf{5}$

$F(T_4) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 1 = 3$

$F(T_5) = 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 0 = 6$

$F(T_6) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = \mathbf{5}$

$F(T_7) = 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 = 2$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P			
P	1	0	1

T									
T	1	0	1	0	1	1	0	1	0

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

P occurs in T at position s
⇔
$F(P) = F(T_s)$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P			
P	1	0	1

T									
T	1	0	1	0	1	1	0	1	0

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_2) =$

P occurs in T at position s
⇔
$F(P) = F(T_s)$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i] \qquad F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P			
P	1	0	1

T									
T	1	0	1	**0**	1	1	0	1	0

$F(P) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_1) = 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

$F(T_2) = (F(T_1) - 2^2 \cdot \mathbf{1}) \cdot 2 + 2^0 \cdot \mathbf{0} = 2$

P occurs in T at position s
⇔
$F(P) = F(T_s)$

Rabin-Karp

$$F(P) = \sum_{i=1}^{m} 2^{m-i} P[i]$$

$$F(T_s) = \sum_{i=1}^{m} 2^{m-i} T[s+i-1]$$

P			
	1	0	1

T									
	1	0	1	0	1	1	0	1	0

F(P) = $2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

F(T_1) = $2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1 = 5$

F(T_2) = (F(T_1) - $2^2 \cdot 1) \cdot 2 + 2^0 \cdot 0 = 2$

F(T_3) = (F(T_2) - $2^2 \cdot 0) \cdot 2 + 2^0 \cdot 1 = 5$

P occurs in T at position s
⇔
F(P) = F(T_s)

Rabin-Karp

- Can compute F(T_{s+1}) from F(T_s):

$$F(T_{s+1}) = 2 \cdot F(T_s) - 2^m T[s] + T[s+m+1]$$

- m large: Numbers too big to calculate in constant time.
- Solution: randomization. Choose prime $p \leq n^2 m$ randomly.

$$F_p(P) = F(P) \mod p = \sum_{i=1}^{m} 2^{m-i} P[i] \mod p$$

$$F_p(T_s) = F(T_s) \mod p = \sum_{i=1}^{m} 2^{m-i} T[s+i-1] \mod p$$

Rabin-Karp

- Can compute $F_p(T_{s+1})$ from $F_p(T_s)$ in constant time:

$$F_p(T_{s+1}) = 2 \cdot (F_p(T_s) \mod p) - (2^m \mod p) \cdot T[s] + T[s+m-1] \mod p$$

- P matches T at position $s \Rightarrow F_p(P) = F_p(T)$.
- Opposite not true.
  - p random prime $\leq n^2 m \Rightarrow$ probability of false match $\leq 2.53/m$.

Rabin-Karp

- Rabin-Karp:
  - Choose random prime ≤ $n^2 m$.
  - Compute $F_p(P)$.
  - For each position s in T compute $F_p(T_s)$ and compare to $F_p(P)$. If $F_p(P) = F_p(T_s)$ declare probable match or check explicitly.
- Time: Θ(m + n) randomized Monte Carlo algorithm (with errors).
- Can verify *all* candidate matches in O(n) time.
  - Las Vegas algorithm (no errors, expected running time) with expected running time O(n):
    - Run algorithm
    - Verify
    - Rerun if errors.