12. String Matching

Goals

- Understand inefficiency in naïve string matching
- Learn string matching using automata
- Learn Rabin-Karp algorithm
- (Knuth-Morris-Pratt algorithm, Boyer-Moore algorithm)

String Matching

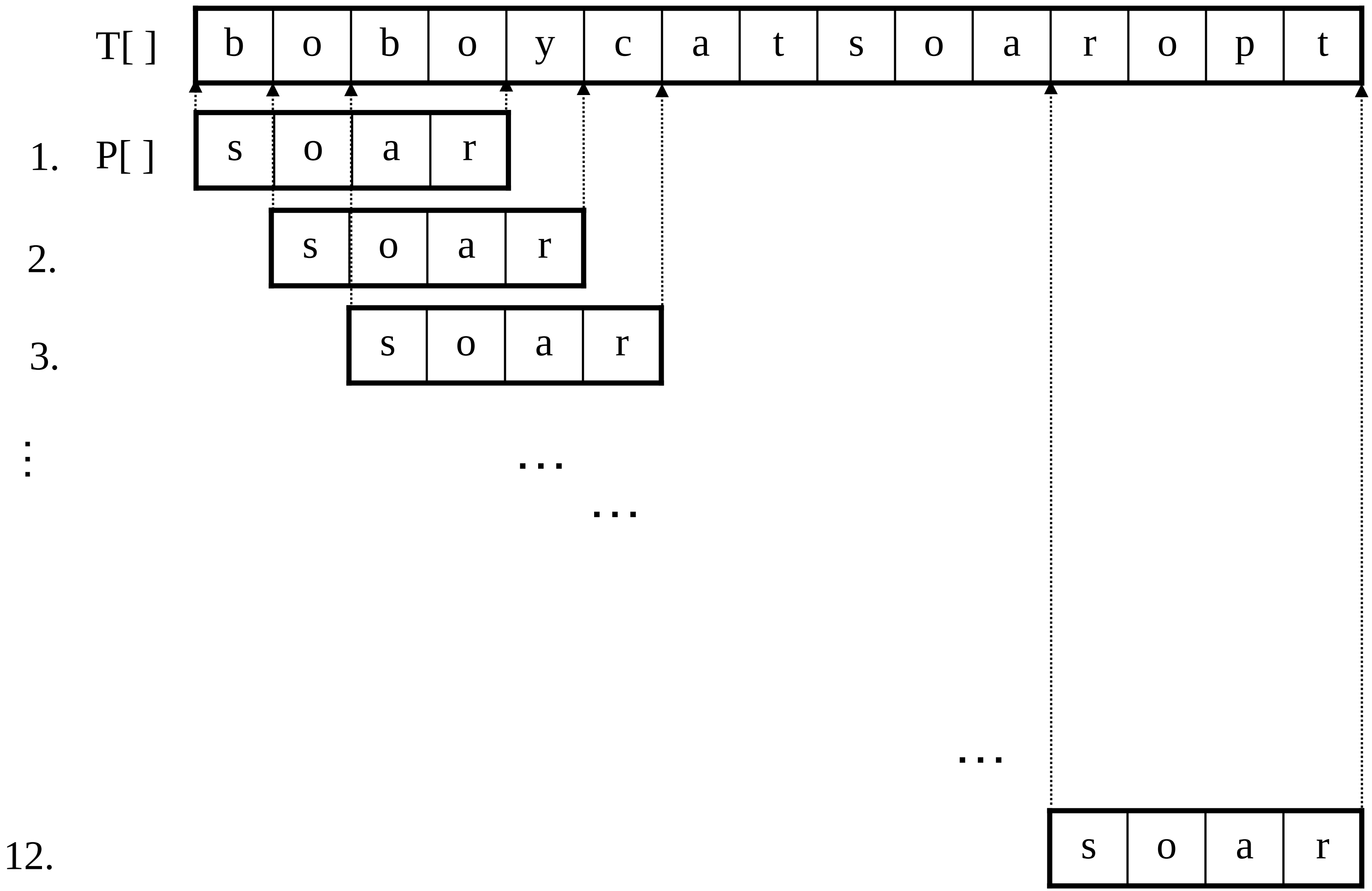
- Input
  - T[1…n]: text string
  - P[1…m]: pattern string
  - $m \ll n$
- String matching problem
  - Find all occurrences of pattern P[1…m] in text T[1…n]

Naïve Algorithm

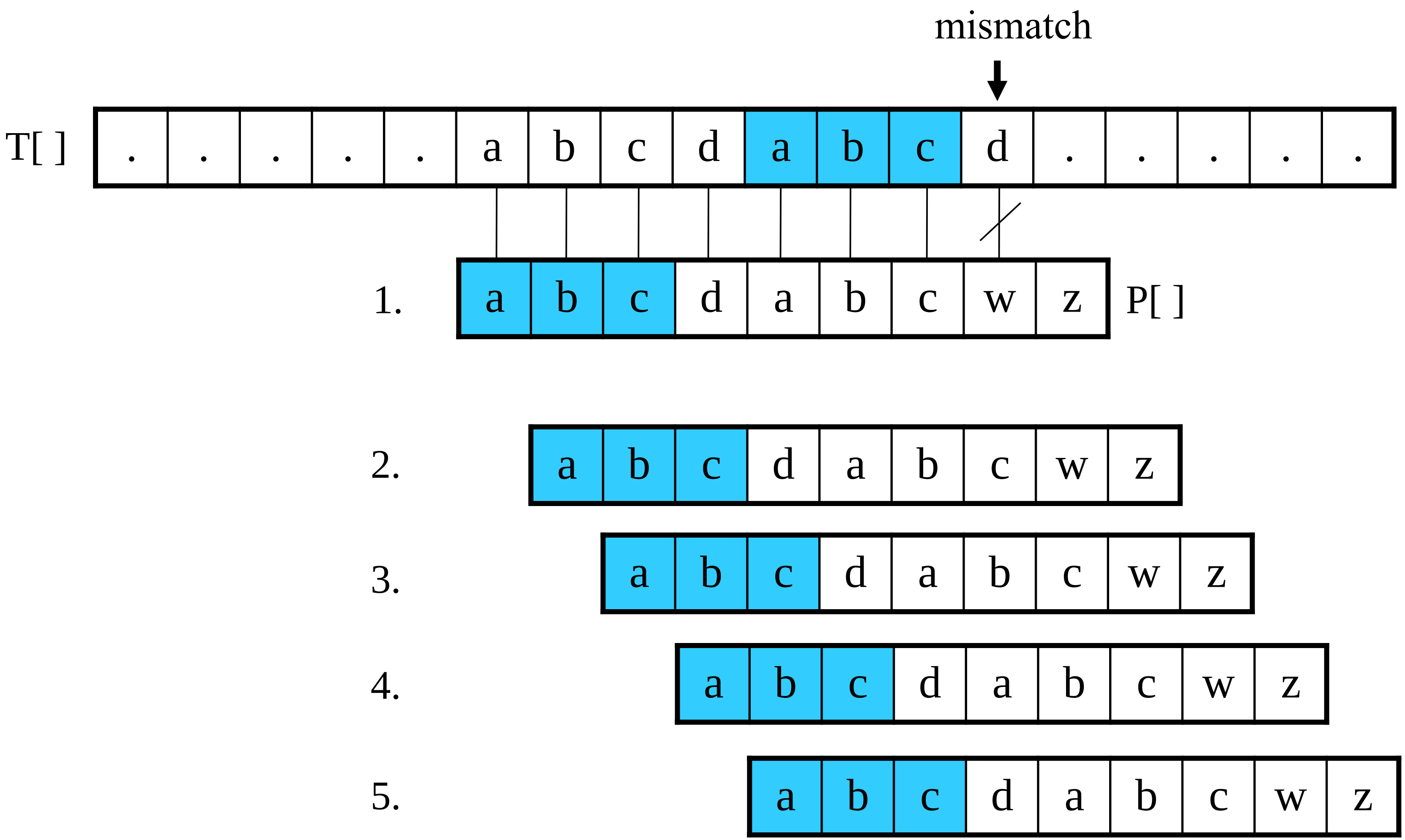
naiveMatching(T, P)
{
    ▷ T[1…n], P[1…m]
    for $i \leftarrow 1$ **to** n-m+1 {
        if (P[1…m] = T[i…i+m-1])
            then output "occurrence at i"
    }
}

✓ Time complexity: $O(mn)$

Naïve Algorithm



Naïve Algorithm



Matching using Automata

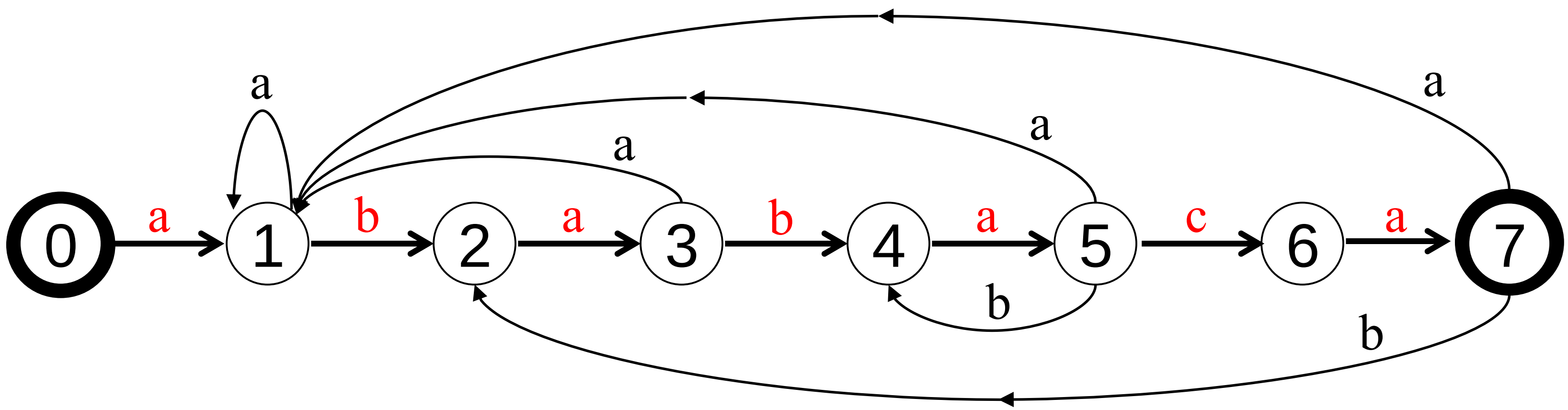
- Automaton: $(Q, q_0, A, \sum, \delta)$
  - Q : set of states
  - q_0 : start state
  - A : set of final states
  - $\sum$: input alphabet
  - δ : state transition function
- Matched characters so far in string matching are represented by states in an automaton

Automaton for Pattern ababaca

Q = {0, 1, … , m}, $q_0 = 0$, $q_f = m$

State i means that P[1…i] matches text characters so far

$\delta(i, a)$ = max{k : P[1…k] is a suffix of P[1…i]a}



Unspecified edges go to state 0

T: dvganbbactababaababacabababacaagbk…

Implementation of Automata

state \ input character	a	b	c	d	e	…	z
0	1	0	0	0	0	…	0
1	1	2	0	0	0	…	0
2	3	0	0	0	0	…	0
3	1	4	0	0	0	…	0
4	5	0	0	0	0	…	0
5	1	4	6	0	0	…	0
6	7	0	0	0	0	…	0
7	1	2	0	0	0	…	0

state \ input character	a	b	c	others
0	1	0	0	0
1	1	2	0	0
2	3	0	0	0
3	1	4	0	0
4	5	0	0	0
5	1	4	6	0
6	7	0	0	0
7	1	2	0	0

Matching using Automata

FA-Matcher (T, δ , f)
▷ f : final state
{
    ▷ T[1…n]
    $q \leftarrow 0$;
    for $i \leftarrow 1$ **to** n {
        $q \leftarrow \delta(q, \mathrm{T}[i])$;
        if ($q = f$) **then** output "occurrence at i-m+1";
    }
}

✓ Time complexity: $\Theta(n + |\Sigma|m)$

Computing Automata

Compute-FA (P, δ , f)

$\triangleright$ P[1…m]

for $i \leftarrow 0$ **to** m

for each $a \in \Sigma$

$k \leftarrow \min(i+1, m)$

while (P[1…k] is not a suffix of P[1…i]a)

$k \leftarrow k - 1$

$\delta(i, a) \leftarrow k$

return δ

✓Time complexity: $O(m^3|\Sigma|)$
✓Can be reduced to $O(m|\Sigma|)$

Rabin-Karp Algorithm

- By converting the pattern string to a number, string comparison is replaced by number comparison.
- Conversion
  - Digit system is determined by the size of alphabet ∑
  - ∑ = {a, b, c, d, e}
    - |∑| = 5
    - a, b, c, d, e correspond to 0, 1, 2, 3, 4, respectively
    - String “cad” is converted to $2*5^2+0*5^1+3*5^0 = 53$

Conversion

- Converting $T[i \ldots i+m-1]$
  - $a_i = T[i+m-1] + d(T[i+m-2] + d(T[i+m-3] + d(\ldots + d(T[i]))\ldots)$
  - $\Theta(m)$ time (Horner's rule)
  - $\Theta(mn)$ for whole text $T[1\ldots n]$
  - Not better than naïve algorithm

- Successive computations
  - $a_i = d(a_{i-1} - d^{m-1}T[i-1]) + T[i+m-1]$
  - d^{m-1} is computed in advance
  - 2 multiplications, 2 additions

Matching with Numbers

P[ ]

e	e	a	a	b

$p = 4*5^4 + 4*5^3 + 0*5^2 + 0*5^1 + 1 = 3001$

T[ ]

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_1 = 0*5^4+2*5^3+4*5^2+1*5^1+1 = 356$

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_2 = 5(a_1-0*5^4)+2 = 1782$

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_3 = 5(a_2-2*5^4)+4 = 2664$

...

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_7 = 5(a_6-2*5^4)+1 = 3001$

...

Matching with Numbers

```
basicRabinKarp(A, T, d, q)
{
    ▷ T[1…n], P[1…m]
    p ← 0; a_1 ← 0;
    for i ← 1 to m {            ▷ compute a_1
        p ← dp + P[i];
        a_1 ← da_1 + A[i];
    }
    for i ← 1 to n-m+1{
        if (i ≠ 1) then a_i ← d(a_{i-1} – d^{m-1}A[i-1]) + A[i+m-1];
        if (p = a_i) then output "occurrence at i";
    }
}
```

✓ Time complexity: $\Theta(n)$

Too Large Number

- Number a_i may be too large, depending on Σ and m
  - There may be an overflow if it exceeds word size

- Solution
  - Use modulo operation to limit a_i
  - Instead of $a_i = d(a_{i-1} - d^{m-1}T[i-1]) + T[i+m-1]$,

    use $b_i = (d(b_{i-1} - (d^{m-1} \bmod q)\,T[i-1]) + T[i+m-1]) \bmod q$
  - Choose a big prime as q such that dq fits within one word

Rabin-Karp Algorithm

P[ ]

e	e	a	a	b

$p = (4*5^4 + 4*5^3 + 0*5^2 + 0*5^1 + 1) \bmod 113 = 63$

T[ ]

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_1 = (0*5^4+2*5^3+4*5^2+1*5^1+1) \bmod 113 = 17$

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_2 = (5(a_1-0*(5^4 \bmod 113))+2) \bmod 113 = 87$

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_3 = (5(a_2-2*(5^4 \bmod 113))+4) \bmod 113 = 65$

...

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_7 = (5(a_6-2*(5^4 \bmod 113))+1) \bmod 113 = 63$

...

Rabin-Karp Algorithm

```
RabinKarp(T, P, d, q)
{
    ▷ T[1…n], P[1…m]
    p ← 0; b_1 ← 0;
    for i ← 1 to m {                              ▷ compute b_1
            p ← (dp + P[i]) mod q;
            b_1 ← (db_1 + T[i]) mod q;
    }
    h ← d^{m-1} mod q;
    for i ← 1 to n-m+1{
            if (i ≠ 1) then b_i ← (d(b_{i-1} – hT[i-1]) + T[i+m-1]) mod q;
            if (p = b_i) then
                    if (P[1…m] = T[i…i+m-1]) then
                            output “occurrence at i”;
    }
}
```

✓ average time: $\Theta(n)$

Thank you