Plan

CÔTE D'AZUR
Algorithmics for Biology
Jean-Paul Comet
Complexity
Pat. Matching
Rabin-Karp
Automata
KMP
BM
Graphs
Dyn.Prog.
Sequences

Exact Pattern Matching

Pattern Matching = search for the presence of certain characteristic features in a sequence of elements.

- *Pattern Matching* = when the search is exact
- *Pattern recognition* = when the pattern search is an approximate search (*approchimate*).

Generally, search for a pattern in a linear or tree-like structure.

CÔTE D'AZUR
Algorithmics for Biology
Jean-Paul Comet
Complexity
Pat. Matching
Rabin-Karp
Automata
KMP
BM
Graphs
Dyn.Prog.
Sequences

Pattern Matching Exact

Exact search for a word of length m in a text of length $n \gg m$.

- compare the m letters of the word with the m letters of the text beginning at position j, for $j = 1, \dots n - m + 1$.
- complexity : $O(m \times n)$

```
n = long[T]
m = long[P]
for (s=0, s<n-m; s++):
    if P[0..m-1] = T[s...s+m-1] then
       print ``the pattern is present at position'', s;
```

To improve the algorithm, information from step j or from previous steps must be taken into account at step $j + 1$.

Example :

- Let us consider the pattern $P = ATAAG$
- If the pattern is present in position i, then we can deduce that in position $i + 1$, $i + 2$, $i + 3$ and $i + 4$ the pattern cannot appear.
- If the letter of the text at i is an A, but the pattern is not present in position i, then the pattern cannot be present in position $i + 1$ (perhaps in position $i + 2$).

CÔTE D'AZUR
Algorithmics for Biology
Jean-Paul Comet
Complexity
Pat. Matching
Rabin-Karp
Automata
KMP
BM
Graphs
Dyn.Prog.
Sequences

Rabin Karp algorithm

Rabin Karp algorithm = integer encoding of substrings

- Worst-case execution time : $O((n - m + 1)m)$
- Average execution time good.

Alphabet : Σ — Word on Σ : string of k consecutive characters
$d = |\Sigma|$ — Word on Σ : number written in base d of length k.

Here : $\Sigma = \{0, 1, 2, \dots 9\}$

Pattern `P[1..m]` : we note p its corresponding decimal value.
Text `T[1..n]` : we note t_s the decimal value of the substring `T[s+1...s+m]`

- Computation of p in $O(m)$ Horner's scheme :
  $p = P[m] + 10(P[m-1] + 10(P[m-2] + \dots + 10(P[2] + 10P[1]))\dots)$
- Computation of t_1 in $O(m)$
- Computation of t_{s+1} : $t_{s+1} = 10(t_s - 10^{m-1}T[s]) + T[s+m]$.
  Example : text $\equiv 134512$ $m = 5$
  $t_1 = 13451$ $t_2 = (13451 - 10^4 \times 1) \times 10 + 2 = 34512$.
  Computation of constant 10^{m-1} in $O(log_2(m))$
  $\Longrightarrow t_0, t_1, \dots t_{n-m}$ can be theoretically computed in $O(n + m)$
  $\Longrightarrow$ occurrences of P in $T[1..n]$ can be computed in $O(n + m)$

Rabin Karp algorithm : Problems

Problem #1

If one of the numbers $t_0, t_1, \dots t_{n-m}$ is too large, arithmetic operations on m digits no longer take a constant time.

> Remedy : computing modulo q.
> p, t_0, t_1 … t_{n-m} modulo q can be computed in $O(n+m)$.
> We choose q such that $10 \times q$ (in fact $d \times q$) just fits on a machine word. The formula for calculating t_{s+1} becomes :
>
> $$t_{s+1} = (10(t_s - T[s]10^{m-1} \mod q) + T[s+m]) \mod q$$

Problem #2

Even if $t_s = p$ implies the equality $(t_s \mod q) = (p \mod q)$, the fact of having calculated the values of t_s modulo q, makes the test insufficient :

- $(t_s \mod q) = (p \mod q)$ does not imply $t_s = p$

> Remedy : We then use $(t_s \mod q) = (p \mod q)$ as a quick test, and when the moduli are equal, we test each letter.

CÔTE D'AZUR
Algorithmics for Biology
Jean-Paul Comet
Complexity
Pat. Matching
Rabin-Karp
Automata
KMP
BM
Graphs
Dyn.Prog.
Sequences

Rabin Karp algorithm : pseudocode

```
1  function Rabin-Karp(Text,Pattern,base,q):
2     n  = len(Text)
3     m  = len(Pattern)
4     h  = base^(m-1) % q    # modulo
5     p  = 0
6     ts = 0
7     for i in [1, m]:       # m included
8        p  = (base * p  + Pattern[i]) % q;
9        ts = (base * ts + Text[i])    % q;
10
11    if (p==ts) : % Test for the first position
12      if (Pattern[1..m] = Text[1..m]):
13        print(``Pattern present at position '', 1);
14
15    for s in [2..n-(m-1)]:    # n-(m-1) included
16      ts = (base*(ts - Text[s]*h)+ Text[s+m]) % q;
17      if (p==ts):
18        if (Pattern[1..m] = Text[s..s+m-1]):
19          print(``Pattern present at position '', s)
```


CÔTE D'AZUR
Algorithmics for Biology
Jean-Paul Comet
Complexity
Pat. Matching
Rabin-Karp
Automata
KMP
BM
Graphs
Dyn.Prog.
Sequences

Rabin Karp algorithm : complexity

Worst-case complexity calculation :

- Choose an example where you spend the whole time comparing letter to letter.
- Example : $P = a^m$ and $T = a^n$

$\Rightarrow$ $\Theta((n-m+1) \times m)$. (quadratic)

Estimated chance of having $t_s = p \mod q$: $\frac{1}{q}$.
In fact, we have one chance in q of choosing $p \ mod \ q$.

Average complexity calculation :

- Lines 2-13 (computation of p,t_1) : $0(m)$
- Lines 15-19 (without any letter-to-letter test) : $O(n)$
- Lines 17-19 (letter-to-letter tests) : $O(m(v + \frac{n}{q})) = O(mv + \frac{n}{q} \times m)$ where v is the number of occurrences of the pattern.

$\Rightarrow$ Average-case complexity : $0(m + n + m(v + \frac{n}{q}))$

> If v is small ($O(1)$ i.e. $\sim$ one occurrence of the pattern) and if $q > m$:
> Average-case complexity : $O(m+n)$.