形式言語とオートマトン

Note6　オートマトンの応用例
2020.5.12作成
2020.6.23update　7.6　2021.6.8　6.27 6.29

中野眞一　群馬大学

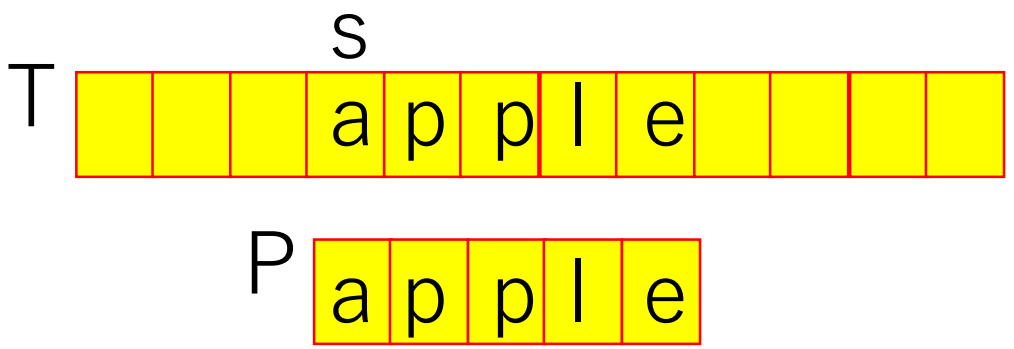
文字列のマッチング

膨大な長さの文字列の中から、指定された文字列を高速に探索する問題を考えよう。
(エディタや、DNA文字列の処理プログラム等で必要とされる。)
この問題を次のように定式化する。

文字列マッチング問題

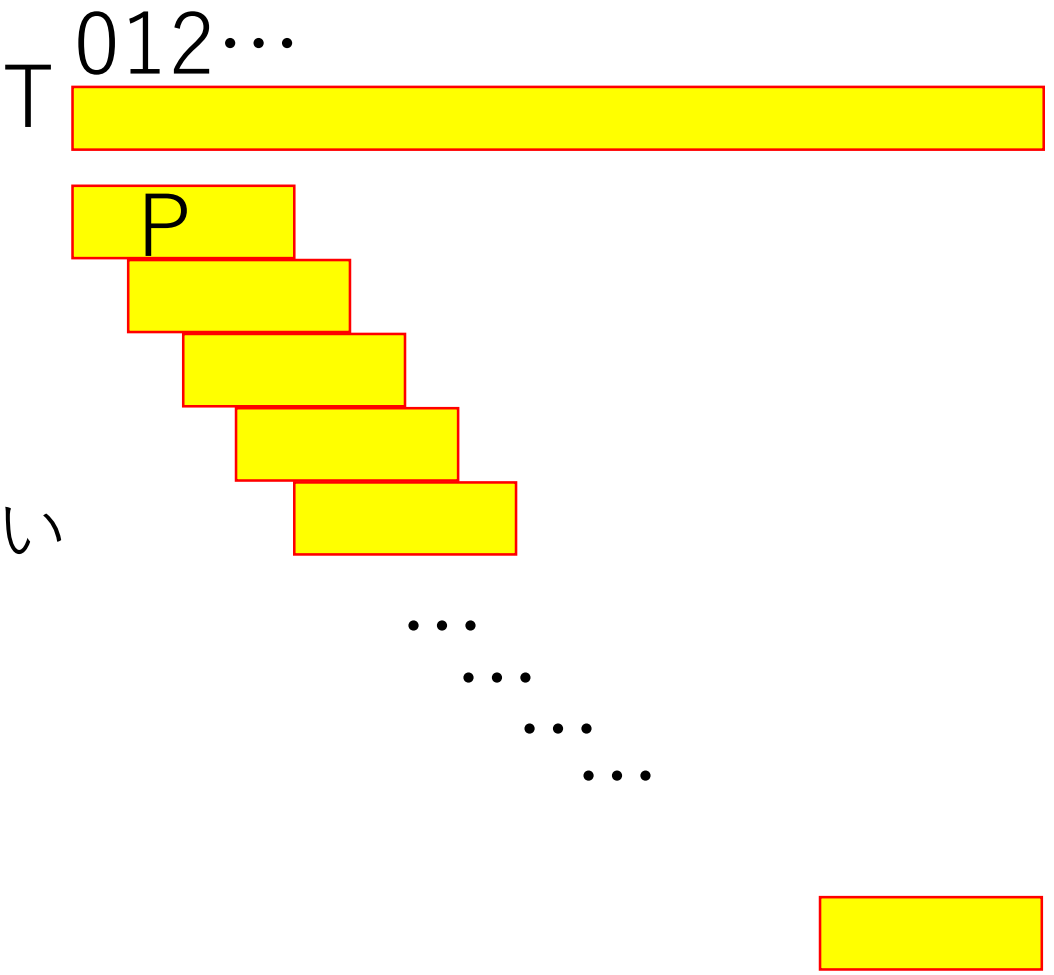
膨大な文字列をテキストTと呼び、文字の配列 T[0..n-1] に格納されているとする。
探索したい文字列をパターンPと呼び、文字の配列 P[0..m-1] に格納されているとする。
m ≦ n とする。T もP も、アルファベット Σ 上の語(文字列)であるとする。

もし、各 j=0,1,2,...,m-1 について、
T[s+j] = P[j] であるならば、
パターンP は、位置s に現れたという。
(matchするという。)



簡単なアルゴリズム　(力づく法)

```
Brute-Force-String-Matcher(T,P)
    for s = 0 to n-m
        for i = 0 to m-1
            if T[s+i] が P[i] と等しくない
            then break
        if i = m
        then s を出力
```



計算時間は O((n-m+1)m)

Rabin-Karp法

剰余(あまり)を上手に利用する方法である。
各文字に、0,1,2,...,|Σ|-1 の数字を割り当てる。
パターンP をm桁の|Σ|進数とみなす。この値をp とする。
テキストT のs文字目から連続するm文字を T_sとし、これをm桁の|Σ|進数とみなし、値をt_s とする。

$T_s = P$ ならば、T中の位置sに、Pが出現したことになる。
このとき、$t_s = p$ となっているはずである。

(問題1)　しかし、t_s や p の値を求めるのに時間がかかる。
簡単な方法では、t_s を1つ計算するのに、O(m)時間かかる。
例　　|Σ| = 10 のとき、
　　p = P[m-1] + 10(　P[m-2] + 10　(P[m-3] + ...　+　10　(P[1] + 10 P[0])　...)) である。

(問題2)　　また、t_s や p の値は、非常に大きくなるかもしれない。(long型でも不足するかも。)

Rabin-Karp法 (つづき)

問題1を解決しよう。各 t_sを独立に計算することをやめ、
t_sから、t_{s+1}を求めることにより、高速化しよう。
例　　$|\Sigma| = 10$ のとき、
　　t_s = 　　$T[s+m-1] + 10(T[s+m-2] + 10(T[s+m-3] + \ldots + 10(T[s+1] + 10\ T[s])\ldots))$ である。
　　$t_{s+1} = T[s+m] + 10(T[s+m-1] + 10(T[s+m-2] + 10(T[s+m-3] + \ldots + 10(T[s+1])$　　である。
これより、　$t_{s+1} = 10\ (t_s - 10^{m-1}T[s]) + T[s+m]$　となる。この式は O(1)時間で計算できる。
(ただし、もし、10^{m-1} の値が計算してあれば。)

例えば、$T = 3141592, m = 5$ のとき、
$t_0 = 31415$ であり、
$t_1 = 10\ (31415 - 10^4 \cdot 3) + 9 = 14159$ である。

上記により、最初に、O(m)時間で、p と t_0を求める。
あとは、1個あたりO(1)時間で、$t_1, t_2, \ldots, t_{n-m}$をそれぞれ求めることができる。

アルゴリズム全体で、 $O(m) + O(n-m+1) = O(n-m+1)$ 時間となる。

Rabin-Karp法 (つづき)

問題2を解決しよう。

適切に数qを選び、qの剰余のみの計算で、上記をすべて実行すればよい。
qは、|Σ|qが、計算機の1 word におさまるような素数とするのが一般的である。

ただし、下記の注意が必要である。
もし、 $t_s \neq p \pmod q$ ならば、PはTの位置sに出現しない。
しかし、$t_s = p \pmod q$ ならば、PはTの位置sに出現するとはいえない。
(まだどちらか不明です。)
(値ではなく、剰余で判定しているので。)

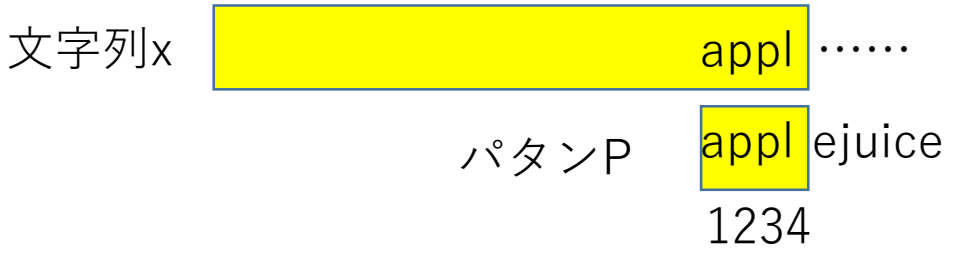
しかし、qを十分大きく選べば、$t_s = p \pmod q$ なのに、
PはTの位置sに出現しないという場合は、ほとんどおこらないとみなせる。

Rabin-Karp法 (つづき)

RK-String-Matcher(T,P,d,q)　　{ d = | Σ | とする }
h = d^{m-1} mod q
p = 0
t_0 = 0
　for i = 0 to m-1　{前処理}
　　p = (d・p + P[i])　mod q
　　t_0 = (d・t_0 + T[i]) mod q
for s = 0 to n-m
　　if p = t_s
　　then　　{本当にmatchかどうか確認する}
　　if P[0..m-1] = T[s..s+m-1]　{ループが必要}
　　then　sを出力
　　if s < n-m
　　then
　　t_{s+1} = (d(t_s - T[s] h) + T[s+m])　mod q

前処理時間は　O(m) 時間である
他の計算時間は O((n-m+1)m)　時間である

最悪の場合は力づく法と同じ計算時間であるが、平均的・実用的には、こちらの方が早いことが多い。



有限オートマトン法

有限オートマトンに基づくアルゴリズムを与える。
この方法は、各文字を、ちょうど1回づつしかチェックしない。
パタンの前処理のほかに、かかる時間は、わずかに O(n)時間である。
(しかし、Σが大きい時には、前処理にかかる時間も大きくなるので注意。)
各パタンPが与えられたとき、このパタンの出現をみつけるオートマトンを設計しよう。

パタンPが与えられたとき、suffix関数σ(シグマ)を次のように定義する。

σ は、Σ*から{0,1,2,...,m}への関数である。xはΣ上の語とする。

$\sigma(x) = \max\{ k \mid P_k \text{ はx の接尾語 } \}$ と定義する。

語xの最後の方に、パタンの最大何文字目までが現れているかを表している。

ここで、P_kは、Pのk文字までの接頭語とする。

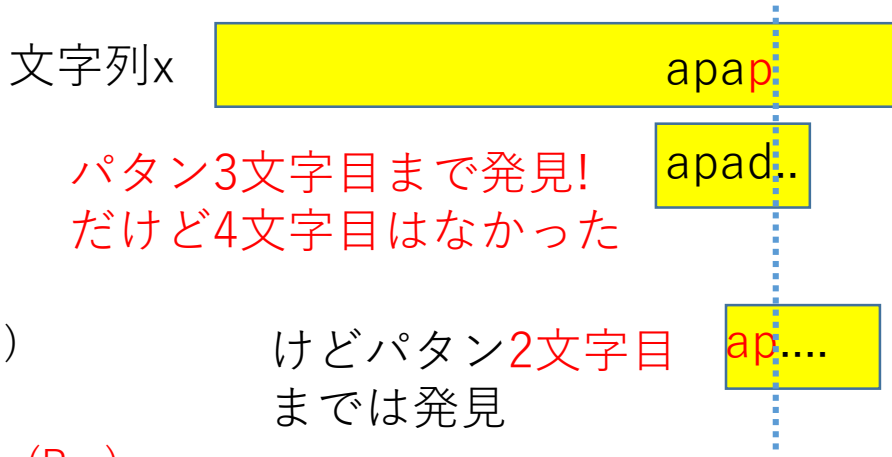
(文字列xの最後とパタンPの最初を、最大、どのくらい重ねられるか。)

有限オートマトン法(つづき)

オートマトンを設計する。
状態 Q = { 0, 1,..., m }とする。(状態 i は、パタンの最初のi文字まで発見!の意味です.)
アルファベット Σ は文字の集合とする。
状態遷移関数を次のように定義する。　$\delta(q,a) = \sigma(P_q a)$
(右辺のかっこの中は、パタンPのq文字目までに文字aを追加した文字列である。)
初期状態 $q_0 = 0$ とする。
受理状態 F = { m } とする。
このオートマトンは次の性質を持つ。
テキストTをi文字目までチェックしたとき、状態がqであるとしよう。
　このとき、$q = \sigma(T_i)$ となる。
　ここで、T_iは、Tのi文字までの接頭語とする。
　(つまり、Tのi文字目までチェックしたとき、Pのq文字目までが出現している!)
もし、次の文字T[i+1]が、Pのq+1文字目と同じならば、状態はq+1へ
もし、次の文字T[i+1](これをaとする)が、Pのq+1文字目と違うならば、状態は $\sigma(P_q a)$ へ
(つまり、Pの出現は無かったが、“次の”チャンスを計算しようとしているのです。
次のチャンスまではスキップ！できる！)
(状態 $\sigma(P_q a)$ の計算にPは関係あるが、Tは関係ないことに注意しよう。)

12345文字目
パタン apadd

パタン4文字目は違った



パタン3文字目まで発見!
だけど4文字目はなかった

けどパタン2文字目
までは発見

有限オートマトン法(つづき)

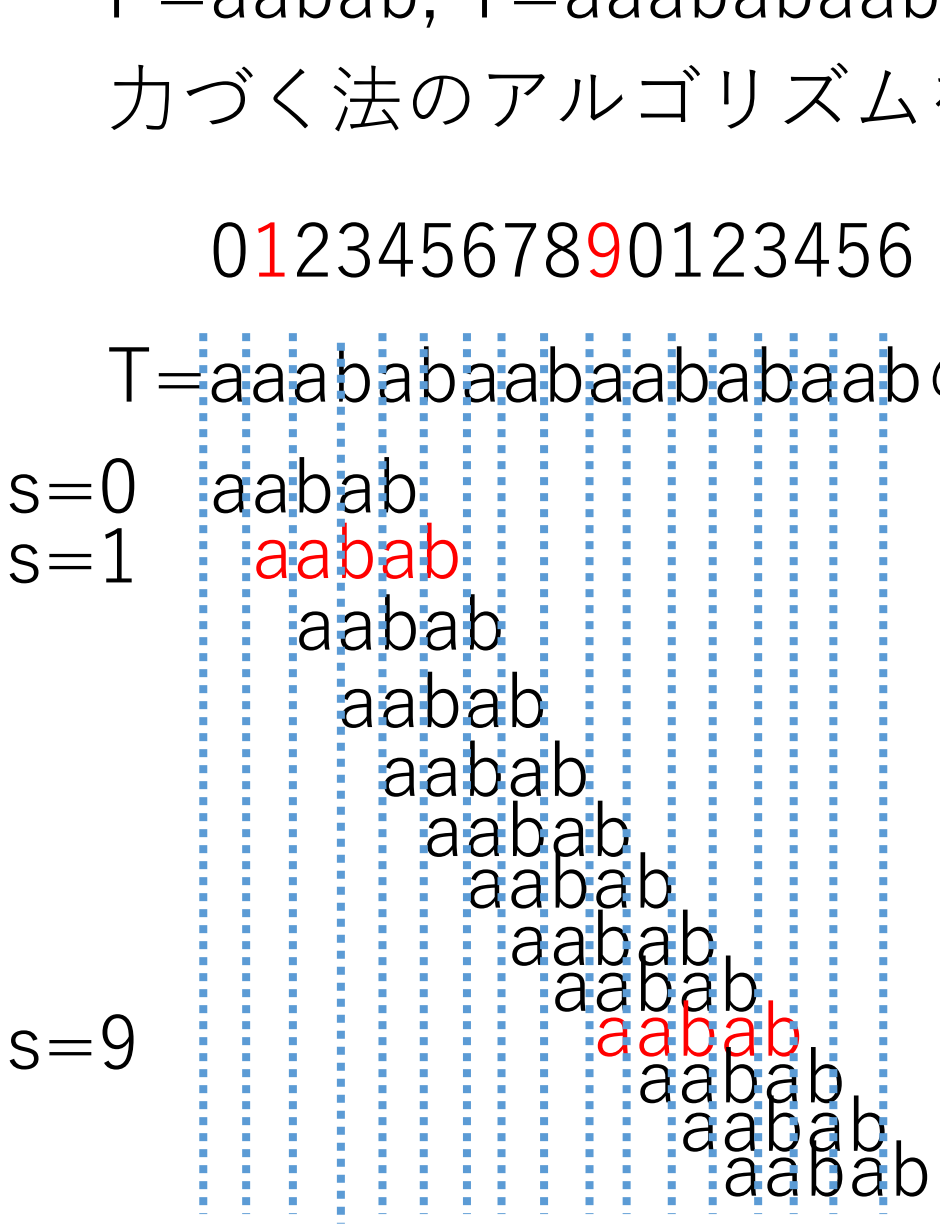
```
FA-String-Matcher(T,P)
{前処理で オートマトンを計算する}
    q = 0
    for i = 0 to n-1
            q = δ(q, T[i])      /* オートマトンを動かしているだけ*/
            if q = m
            then    i-m を出力
```

前処理時間は　O(m | Σ |)
matching 時間は O(n)

理解確認クイズ

(1)
P=aabab, T=aaababaabaababababaabのとき,
力づく法のアルゴリズムを説明せよ。

P=aabab, T=aaababaabaababababaabのとき，
力づく法のアルゴリズムを説明せよ。



S=0,1,2,3,…の位置に
Pがあるかをすべて調べる。
S=1およびS=0の位置に
Pがあるのでこれを出力する

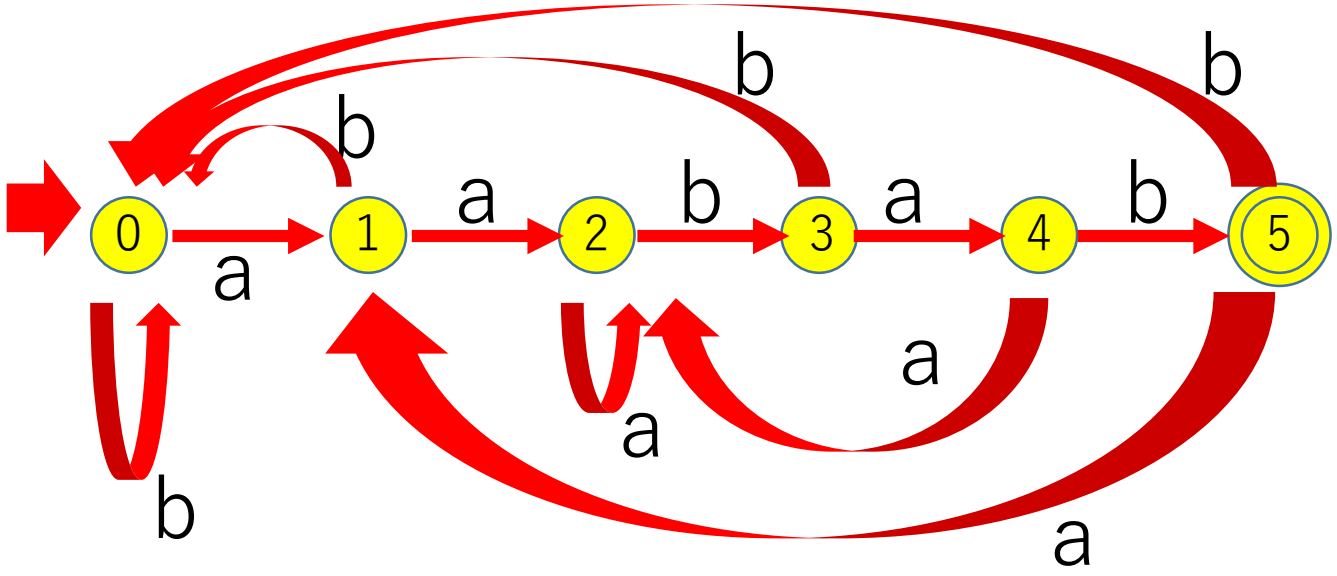
理解確認クイズ

(2)
P=aabab, T=aaababaabaababababaabのとき,
有限オートマトン法のアルゴリズムを説明せよ。

理解認識クイズ

T　aaababaabaababababaab
状態 0122345123423451 23

(2) P=aabab, T=aaababaabaababababaabのとき,
有限オートマトン法のアルゴリズムを説明せよ。



状態2からの遷移
aaa　状態2へ
aab　状態3へ

理解確認クイズ

(3)
P=ababaca, T=abababacabaのとき,
有限オートマトン法のアルゴリズムを説明せよ。

オートマトン設計してみましょう