アルゴリズムとデータ構造
第14回: 文字列照合

担当: 上原 隆平(uehara)

2014/05/29

文字列照合 (String matching)

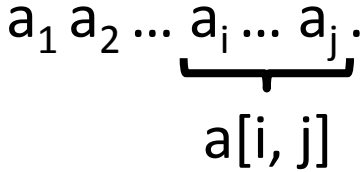
- テキスト文字列: text = $t_1 t_2 \ldots t_n$
- パターン文字列: patt = $p_1 p_2 \ldots p_m$
- 問題: text の中に patt と等しいものが含まれるか. 含まれるときはその位置を出力せよ.
- 例:
  - text = ababcababcabacbab
  - patt = abcabac
  - 答: ababcababcabacbab (8)

Q. どうやって計算する?

素朴な方法: 逐次探索

- 入力
  - t[]: 1番目からn番目
  - p[]: 1番目からm番目
- 出力:
  - s if t[s, s+m-1] = p[1, m]
  - 0 otherwise

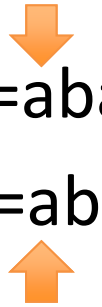
a[s, e]: s番目からe番目までの部分列, e.g.,




```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

素朴な方法: 逐次探索

- 例:
  - text=ababcababcabacbab
  - patt=abcabac



不一致

```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

素朴な方法: 逐次探索

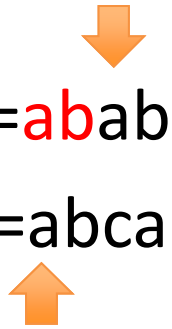
- 例:
  - text=ababcababcabacbab
  - patt=abcabac

不一致

```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

素朴な方法: 逐次探索

- 例:
  - text=ababcababcabacbab
  - patt=abcabac

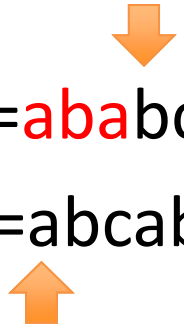


不一致

```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

素朴な方法: 逐次探索

- 例:
  - text=ababcababcabacbab
  - patt=abcabac

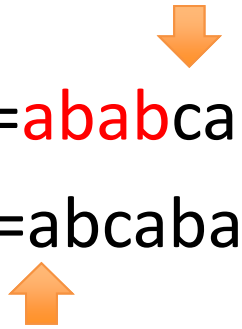


不一致

```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

素朴な方法: 逐次探索

- 例:
  - text=ababcababcabacbab
  - patt=abcabac

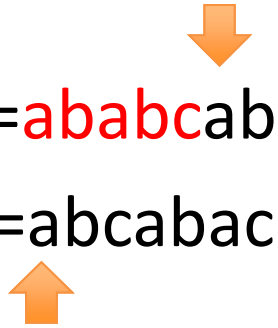


不一致

```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

素朴な方法: 逐次探索

- 例:
  - text=ababcababcabacbab
  - patt=abcabac

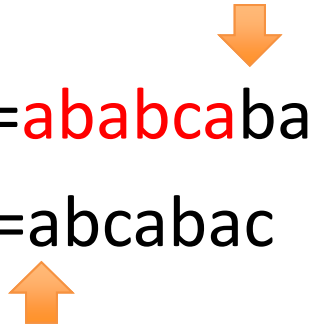


不一致

```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

素朴な方法: 逐次探索

- 例:
  - text=ababcababcabacbab
  - patt=abcabac

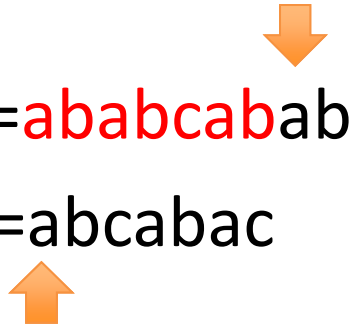


不一致

```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

素朴な方法: 逐次探索

- 例:
  - text=ababcababcabacbab
  - patt=abcabac



一致発見

```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

素朴な方法の計算時間

- 最悪の場合:
  for $1 \leqq s \leqq n-m+1$,
  $t[s, s+m-2] = p[1, m-1] \ \wedge \ t[s+m-1] \neq p[m]$

- 例:
  - text = aaaaaaaaaa
  - patt = aaab

- 計算量: O(m n)

```
for(s=1;s<=n-m+1;s++){
  for(i=1;i<=m;i++)
    if(t[s+i-1]!=p[i])
      break;
  if(i == m+1)
    return s;
}
return 0;
```

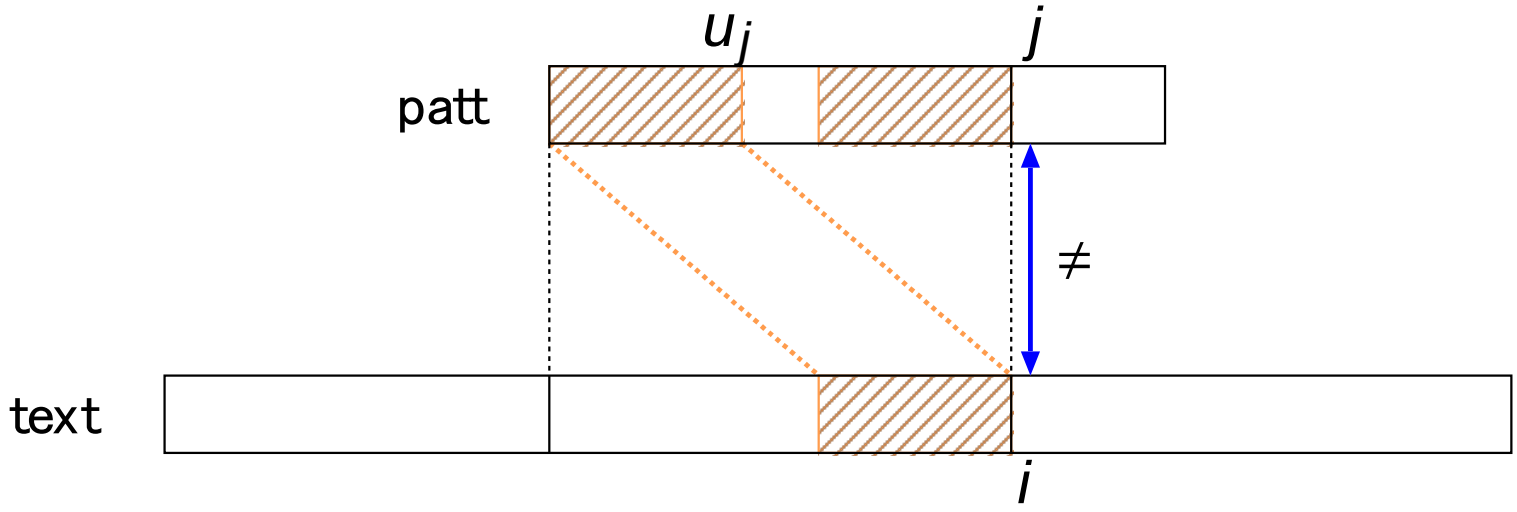
素朴な方法が遅い理由

- テキストとパターンの不一致が起こるたびに，
  テキストを指すポインタが前に戻っている

- Q: 戻らないようにできるか？
- A: できる
  - t[i] で不一致が起きた場合は，
    次の照合は t[i] から始める（戻らない！）
  - p[] の添字はうまくずらす

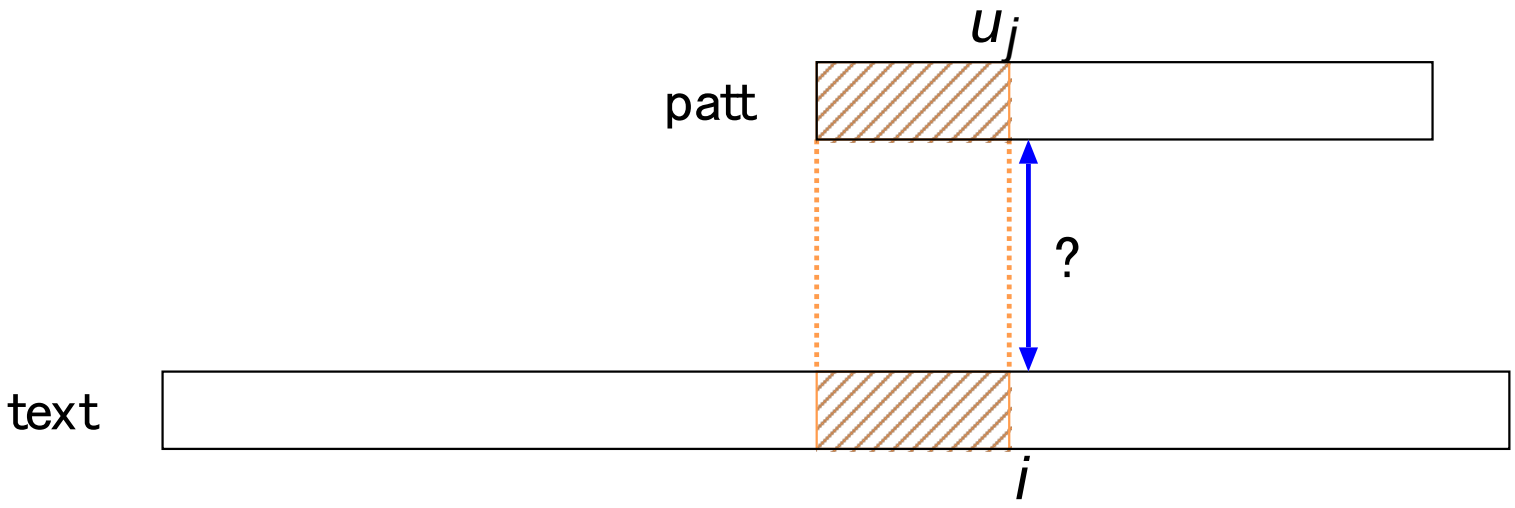
素朴な方法の改善

- t[i-j+1, i-1] = p[1, j-1] かつ t[i] ≠ p[j] とする
- 次に比べるのは t[i] と $p[u_j+1]$ から
  - u_j: p[1, k] = p[j-k, j-1] となる最大の k < j-1
    - 1文字も一致しなければ 0
- $p[u_j+1]$: t[i] とマッチする可能性のある文字で，最も右にあるもの



素朴な方法の改善

- t[i-j+1, i-1] = p[1, j-1] かつ t[i] ≠ p[j] とする
- 次に比べるのは t[i] と $p[u_j+1]$ から
  - u_j: p[1, k] = p[j-k, j-1] となる最大の k < j-1
    - 1文字も一致しなければ 0
- $p[u_j+1]$: t[i] とマッチする可能性のある文字で，最も右にあるもの





KMP (Knuth-Morris-Pratt) 法

- u_j の値が前もって計算されているとき
  - text[]: 1..n,　patt[]: 1..m

```
i=1; j=1;
while (j<=m && i<=n)
  if(j==0 || text[i]==patt[j])
    i++; j++;
  else
    j=u[j];
if(j==m+1)
  パターンをテキストのi-m番目からの部分に発見;
```

KMP法: u_j の計算

- u_j の計算でもパターン照合を行う
- patt の j 文字目まで調べたとき
  - u_k の値が k = 1,…,j まで計算されていると仮定
  - 照合に必要なのは k ≦ j についての u_k

```
k=0; j=1; u[1]=0;
while(j < m){
  if(k==0||patt[k]==patt[j])
    k++; j++; u[j] = k;
  else
    k = u[k];
}
```

ほぼ一緒

```
i=1; j=1;
while (j<=m && i<=n)
  if(j==0 || text[i]==patt[j])
    i++; j++;
  else
    j=u[j];
```

KMP法

KMP法: 計算時間

```
i=1; j=1;
while (j<=m && i<=n)
  if(j==0 || t[i]==p[j])
    i++; j++;
  else
    j=u[j];
```

- j の減少回数 < j の増加回数
  - 0でないときのみ減少する
- j の増加回数 = i の増加回数
- i の増加回数: n

- 全体の計算時間 < 3n ∈ O(n)

KMP法: u_jの計算込みの全体の計算時間

- u[]の構築はKMP法と変わらない
  ➔ O(m)
  - mはpattの長さ
- よってu[]を構築して文字列照合を行うと
  O(n + m)
  - nはtextの長さ

ミニ演習

- 以下の例に対してKMP法を実行せよ
  - text = aaaaaaaaaa
  - patt = aaab