김일성종합대학학보 주체109(2020)년
수 학 제66권 제1호

뒤붙이나무에 의한 k－근사문자렬대조의 고속화를 위한 한가지 알고리듬

리지성, 조영선

선행연구[2]에서는 문자렬사이의 편집거리를 계산하기 위한 동적계획법은 k－근사문자렬대조를 진행하면서 그 위치까지 찾아낼수 있는 효과적인 방법으로 되고있으며 검색시간은 $O(mn)$이라는것을 밝혔다.

선행연구[4]에서는 뒤붙이나무를 리용하여 매 가지에서 k－근사문자렬대조를 진행함으로써 시간을 $O(kn)$으로 줄인 알고리듬이 제기되였다.

론문에서는 대자료처리분야에서 중요하게 제기되는 근사문자렬대조의 성능을 개선하기 위하여 뒤붙이나무를 리용하여 k－근사문자렬대조를 진행할 때 뒤붙이나무의 매 가지에서 대조속도를 높이는 한가지 알고리듬을 제안하였다.

선행연구[2, 3]에서는 보통 k－근사문자렬대조에서는 매 가지에서 견본문자렬의 길이를 m이라고 할 때 $m+k$개 문자에 대한 대조를 진행하는데 사실 k개이상의 문자렬이 차이나면 더이상 대조할 필요가 없다는것을 밝혔다.

그러므로 매 가지에서의 대조과정에 다른 문자라고 판정되면 불일치를 나타내는 계수기의 값을 증가시키다가 대조위치가 불일치계수기의 값보다 k+1만큼 차이나게 되면 대조를 다음가지에로 넘긴다. 불일치계수기의 값이 $m-k$ 로 되면 k－근사문자렬로 판정할수 있다. 이것을 실현한 알고리듬을 아래에 서술하였다.

입력: 본문 $T=T_1T_2\cdots T_n$과 견본문자렬 $P=P_1P_2\cdots P_n$

출력: 본문에서 견본문자렬과 k만큼 차이나는 문자렬의 위치 *pos*

주어진 본문에 대하여 뒤붙이나무를 구성한다. 우꼬넨의 알고리듬에 의하면 실행시간은 $O(n)$이다. 얻어진 뒤붙이나무의 매 가지에 대하여 다음의 알고리듬에 따라 k－근사문자렬대조를 진행한다. 알고리듬에서 su는 본문문자렬에서 뒤붙이들의 위치, lcp는 린접한 두 뒤붙이들사이의 최대공통앞붙이의 길이이다.

```
while (0 ≤ i ≤ m) do
c[i] ←i                //새로운 동적계획표작성에 리용되는 렬
                       //이전가지에서 작성한 동적계획표의 lcp번째 렬로 초기화
end while
pos =su + lcp;
while (pos ≤ n) do
cnt ← min {pos + k, m}             // cnt: 불일치계수기
d[0] ← pos + 1       // d[i]: 동적계획표의 새로운 렬
i ← 0
while(i ≤ cnt) do              //동적계획표작성단계
if (P[i] = T[pos]) then
d[i+1] ← c[i]
else
```

```
d[i+1] ← min{c[i], d[i], c[i+1]} + 1
end if
end while
c ← d
while (c[cnt] > k) do            //동적계획표에서 k보다 작은 자리찾기
cnt ← cnt - 1
end while
if(cnt = m) then
pos값을 출력          // su~pos까지가 구하려는 문자렬이다.
else
if(pos > cnt + k) then       //대조중지
      다음가지를 탐색
end if
end if
end while
```

실례로 T=TACCCTGGCCTGA, P=GTCA, k=2일 때 알고리듬의 실행과정을 보면 다음과 같다. 먼저 T=TACCCTGGCCTGA의 뒤붙이나무를 구성한다.(그림)

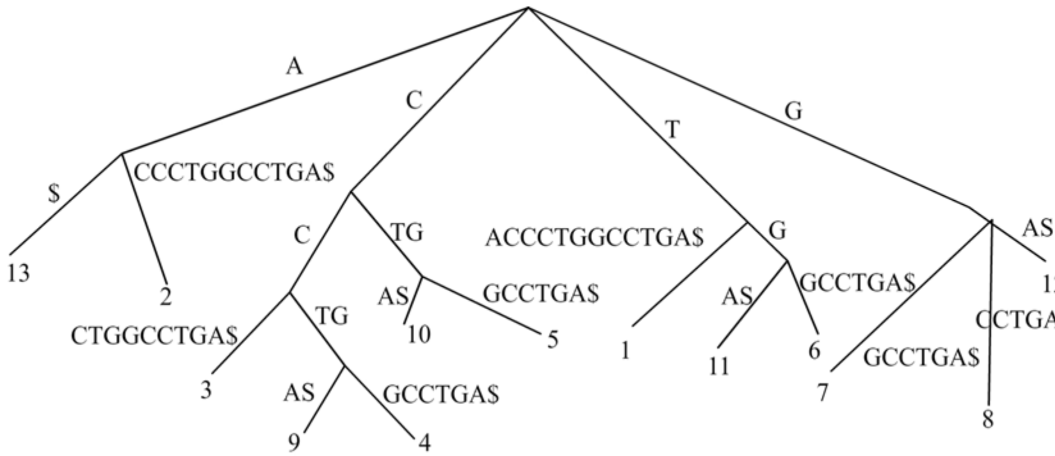

그림. T=TACCCTGGCCTGA의 뒤붙이나무

표. T=TACCCTGGCCTGA, P=GTCA에 대한 동적계획표

		T	A	C	C	C	T	G	G	C	C	T	G	A
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
G	1	1	2	3	4	5	6	6	7	8	9	10	11	12
T	2	1	2	3	4	5	5	6	7	8	9	9	10	11
C	3	2	2	2	3	4	5	6	7	7	8	9	10	11
A	4	3	2	3	3	4	5	6	7	8	8	9	10	10

다음 뒤붙이나무의 매 가지에 대하여 동적계획법을 리용한 k－근사문자렬대조를 진행한다. 이를 위한 동적계획표 $M_{i,j}$를 작성한다.(표) 여기서 제일 아래행을 고찰하여 k보다 크지 않은 위치를 택하면 목적하는 위치를 얻게 된다.

다음으로 알고리듬의 시간복잡도를 평가한다.

보조정리 1 $m \ge j$, $\beta < 1$일 때 길이가 j인 두 우연문자렬들이 길이가 $k < j$인 부분문자렬들을 포함할 확률이 $(1/j^2)\alpha\beta^j$보다 크지 않게 되는 $\alpha = O(m)$이 존재한다.

증명 스털링의 공식[1] $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$으로부터 $cj = k$라고 하면

$$\frac{j!}{(cj)!(j-cj)!} = (1+o(1))(\sqrt{2\pi c(1-c)j} \cdot c^{cj}(1-c)^{(1-c)j})^{-1}$$

이 성립한다.

p를 구하려는 확률, b를 자모의 크기라고 할 때

$$p = \binom{j}{c \cdot j}^2 b^{-cj} \le \frac{m(1+o(1))}{2\pi c(1-c)} \cdot \frac{1}{j^2} \cdot (c^c(1-c)^{1-c})^{-2j} \cdot b^{-cj}$$

을 얻는다.

$$\alpha = \frac{m(1+o(1))}{2\pi c(1-c)},\ \beta = (c^c(1-c)^{1-c})^{-2} \cdot b^{-c}$$

라고 하면 $p \le (1/j^2)\alpha\beta^j$이 성립한다.

이때 $\alpha = O(m)$으로 된다.(증명끝)

보조정리 2 알고리듬에서 $E(cnt^2) = O(m + k^2)$이 성립한다.

증명 $0 < c < 1$에 대하여 $q = 2k/(1-c)$라고 하자. 즉 $\forall j \ge q,\ j - 2k \ge cj$이다.

이때

$$E(cnt^2) = E(cnt)^2 - V(cnt) < (q-1)^2 + \sum_{j \ge q} j^2 P(M_{j,i} \le k)$$

이 성립한다.

보조정리 1에서 m을 견본문자렬의 길이로 주면 $P(M_{j,i} \le k) < (1/j^2)\alpha\beta^j$이며 두 식을 결합하면

$$E(cnt^2) < (q-1)^2 + \sum_{j \ge q} j^2 (1/j^2)\alpha\beta^j = O(k^2) + O(m) = O(m + k^2)$$

이 성립한다.(증명끝)

정리 알고리듬에서 두번째 단계의 평균시간복잡도는 $O(m + k^2)$이다.

증명 알고리듬이 $k + j$위치에서 다음가지에로 넘어간다고 하고 이때 이 사건의 확률을 P_j, 동적계획표에서 갱신된 항들의 개수를 N_j이라고 하자.

매 단계에서 한문자씩 거치는것으로 하여 cnt+1개의 항들이 갱신되며 cnt는 기껏 하나씩 커진다. 이로부터 다음의 결과를 얻는다.

$$N_j \le \sum_{i=1}^{k+j} (k+i) = (3k^2 + k + 4kj + j^2 + j)/2$$

한편

$$E(N_j) \le P_1N_1 + P_2N_2 + \cdots + (P_m + P_{m+1})N_m < P_1N_1 + P_2N_2 + \cdots + P_mN_m + P_{m+1}N_{m+1}$$

이고 알고리듬에서 대조에 참가하는 문자수는 기껏 $m+k$ 이므로

$$E(N_j) = O(k^2) + O(k) + O\left(\sum_{i=1}^{m+1} P_i i^2\right)$$

이다. 보조정리 2로부터 $E(i^2) = O(m+k^2)$ 이고 우의 두 식을 결합하면 $E(N_j) = O(m+k^2)$ 이다.(증명끝)

결과를 종합하면 실행시간이 $O(m+k^2)$ 인것으로 하여 n 의 크기가 큰 대용량본문들에 대하여 대조를 진행할 때 효과적이라는것을 알수 있다.

참 고 문 헌

[1] B. Miklos; A Walk Through Combinatorics, Springer, 132～138, 2004.

[2] Yoshimasa Takabatake et al.; Algorithms, 9, 26, 2016.

[3] Huan Hu et al.; Knowl. Inf. Syst., 49, 121, 2016.

[4] F. Simone; LNCS 9778, 65, AAIM, 2016.

주체108(2019)년 9월 15일 원고접수

An Algorithm for Making High-Speed k − Approximate String Matching over Suffix Tree

Ri Ji Song, Jo Yong Son

In this paper, we propose a k-approximate string matching algorithm that reduces the number of searched characters on each path of suffix tree and uses the dynamic program to reduce its searching time.

Keywords: approximate string matching, suffix tree, dynamic programming