순위다중패턴매칭을 위한 해싱기반 알고리즘의 이동테이블 병렬계산

박정훈, 김영호, 권상훈, 심정섭†
인하대학교 컴퓨터공학과
e-mail: pjh24749297@gmail.com, yhkim8505@gmail.com,
sanghoon3018@gmail.com, jssim@inha.ac.kr

Parellel Computation of the Shift Table of a Hashing-Based Algorithm for the Order-Preserving Multiple Pattern Matching

Jeonghoon Park, Youngho Kim, Sanghoon Kwan, Jeong Seop Sim†
Department of Computer Engineering, Inha University

요 약

길이가 같은 두 문자열의 같은 위치에 있는 문자의 순위가 모두 일치할 때, 두 문자열은 순위동형이라 한다. 순위다중패턴매칭문제는 텍스트 T와 k개의 패턴들의 집합 $P'=\{P_1,P_2...,P_k\}$이 주어졌을 때, P'의 패턴들과 순위동형인 T의 모든 부분문자열의 위치를 찾는 문제이다. 최근 전처리단계에서 P'에 대한 이동테이블을 $O(kmq\log q)$ 시간에 계산하여 순위다중패턴매칭문제를 해결하는 해싱기반 알고리즘이 제시되었다. 이때 P'에서 가장 짧은 패턴의 길이를 m, q-그램의 길이를 q라고 한다. 본 논문에서는 P'이 주어졌을 때, 이동테이블을 $O(mq\log q)$ 시간에 계산하는 병렬알고리즘을 제시한다. 실험결과, 본 논문에서 제시하는 병렬알고리즘은 k개의 스레드를 이용하여 $m=100$, $q=5$에 대해 $k=100$일 때와 $k=1{,}000$일 때 순차알고리즘보다 각각 약 12.9배, 약 215배 빠른 수행시간을 보였다.

1. 서론

순위패턴매칭문제(order-preserving pattern matching problem)는 길이가 각각 m과 n인 문자열 P와 T가 주어졌을 때, P와 각 문자들의 순위가 같은 순서로 발생하는 T의 모든 부분문자열의 위치를 찾는 문제이다. 순위다중패턴매칭문제(order-preserving multiple pattern matching problem)는 k개의 패턴들의 집합 $P'=\{P_1,P_2...P_k\}$과 T가 주어졌을 때, 패턴집합에 속한 $P_i(1\le i\le k)$의 각 문자들의 순위와 같은 순서로 발생하는 T의 모든 부분문자열의 위치를 찾는 문제이다. [1]에서는 순위다중패턴매칭에 대해 Aho-Corasick 오토마타 기반 알고리즘을 제시하였고, [2]에서는 q-그램을 이용한 오문자규칙으로 Wu-Manber 알고리즘을 변형하여 같은 문제를 해결하였다. 이때 P'에 오문자규칙을 적용하기 위한 이동테이블은, P'에서 가장 짧은 패턴의 길이를 m, q-그램의 길이를 q라고 할 때, $O(kmq\log q)$ 시간에 계산된다.

GPU의 성능이 향상됨에 따라 문자열매칭 분야에서도 GPU를 활용한 병렬화 연구가 활발해지고 있다[3,4,5]. [3]에서는 특정조건에서 교환연산을 포함한 확장편집거리문제를 해결하는 병렬알고리즘을 제시하였고, [4]에서는 사각망 순열패턴매칭문제를 해결하는 병렬알고리즘을 제시하였다. [5]에서는 순위다중패턴매칭을 위한 Z-함수를 선형시간에 계산하는 병렬알고리즘을 제시하였다.

본 논문에서는 GPU의 특성을 고려하여 [2]에서 제시된 알고리즘의 전처리단계를 병렬화한 알고리즘을 제시한다. 패턴의 개수만큼 스레드를 할당하여 이동테이블을 $O(mq\log q)$시간에 계산한다. 실험결과, 제시한 병렬알고리즘은 $m=100$, $q=5$에 대해 $k=100$일 때와 $k=1{,}000$일 때 순차알고리즘보다 각각 약 12.9배, 약 215배 빠른 수행시간을 보였다. $m=50$, $q=5$, $k=1{,}000$일 때 병렬알고리즘은 순차알고리즘보다 약 135배 빠른 수행시간을 보였다.

본 논문의 구성은 다음과 같다. 2장에서 용어 정의와 관련 연구를 설명한다. 3장에서는 패턴집합이 주어졌을 때 패턴집합에 대한 이동테이블을 계산하는 병렬알고리즘을 제시한다. 4장에서는 실험을 통해 이동테이블을 계산하는 순차알고리즘과 병렬알고리즘의 수행시간을 비교한다.

* 이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2014R1A2A1A11050337).
* 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단 포스트게놈다부처유전체사업의 지원을 받아 수행된 연구임 (NRF-2014M3C9A3064706).
† 교신저자

2. 관련 연구

2.1 용어 정의

Σ는 정수로 구성된 문자집합을 나타낸다. Σ로 구성된 문자열 x가 주어졌을 때, x의 길이를 $|x|$, x의 i번째 문자를 $x[i]$로 표기한다. x의 i번째부터 j번째까지의 부분문자열을 $x[i...j]$로 표기하고, x의 i번째까지의 접두사 $x[1...i]$를 x_i로 표기한다. Σ로 구성된 패턴들의 집합을 $P'=\{P_1,P_2,...,P_k\}$라고 할 때, P'에 속한 패턴들 중 가장 짧은 패턴의 길이를 m으로 표기한다. 순위동형(order-isomorphic)이란 길이가 같은 두 문자열이 주어졌을 때, 두 문자열의 같은 위치에 있는 문자의 상대적인 순서가 모두 일치하는 것으로 정의된다.

문자열의 순위정보를 표현하는 방법으로 접두사표현법(prefix representation)이 있다. 접두사표현법 $Pre(x)$는 다음과 같이 정의된다.

$$Pre(x)[i]=|\{j:x[j]\le x[i]\ \forall 1\le j<i\}|$$

즉, $Pre(x)[i]$는 $x[1...i-1]$에서 $x[i]$보다 작거나 같은 문자의 개수를 저장한다. 이를 이용하여 x_i에서 $x[i]$의 순위를 알 수 있다. 두 문자열 x와 y가 $|x|=|y|$이고 $Pre(x)=Pre(y)$이면, x와 y는 서로 순위동형이다.

2.2 오문자규칙 및 핑거프린트함수

오문자규칙(bad-character rule)은 Boyer-Moore 알고리즘에서 문자의 일치여부에 대한 조사가 불필요한 구간을 건너뛰기 위해 사용하는 이동규칙이다[6]. $T[i-m+1...i]$와 P의 일치여부를 비교할 때, 후향탐색(backward search) 방식으로 뒤에서부터 앞으로 순서대로 일치여부를 검사한다. T의 i위치에서 불일치가 발생하면 $T[i]$를 오문자라 하며, 이때의 이동거리는 오문자와 일치하는 $P[1...i-1]$의 문자들 중 가장 오른쪽에 위치한 문자를 $T[i]$의 위치로 이동시키는 거리이다.

순위패턴매칭문제에서는 한 문자에 대한 순위는 항상 0이기 때문에 문자열매칭에서의 오문자규칙을 그대로 사용할 수 없다. 따라서 오문자규칙을 순위패턴매칭문제에 적용하기 위해, q개의 문자를 하나의 단위로 취급하는 q-그램(q-gram)과 q-그램 x의 접두사표현법에 의한 순위정보를 하나의 숫자로 변환할 수 있는 핑거프린트(finger print) 함수를 이용한다. 핑거프린트함수 $f(x)$는 다음과 같이 정의된다.

$$f(x)=\sum_{k=1}^{q}Pre(x)[k]\cdot(k-1)!+1$$

$f(x)$의 값 α의 범위는 $1\sim q!$이다. 핑거프린트함수는 순위통계트리(order-statistics tree)를 이용하여 $O(q\log q)$ 시간에 계산될 수 있다[2]. q-그램과 핑거프린트함수를 이용하면 q개의 문자의 순위정보에 대해 오문자규칙을 정의할 수 있다.

2.3 해싱기반 순위다중패턴매칭 알고리즘

[2]에서는 순위다중패턴매칭을 위해 q-그램과 핑거프린트함수를 활용한 오문자규칙과 Wu-Manber 알고리즘 기반의 해싱을 이용한 알고리즘을 제시하였다. 이 알고리즘은 전처리단계와 검색단계로 구성된다. 전처리단계에서는 패턴집합 $P'=P_1,P_2,...P_k$에 대한 해시테이블(hash table), 이동테이블(shift table), 접두사표현법을 나타내는 접두사테이블(prefix table)을 생성한다. 해시테이블은 $P_i(1\le i\le k)$ 중에서 마지막 q-그램 $P_i[m-q+1...m]$의 핑거프린트함수 값이 동일한 패턴들을 리스트로 연결하여 저장한 테이블이다. 검색단계에서 해시테이블을 참조하면, 순위동형 가능성이 있는 후보패턴 P_i를 찾을 수 있다.

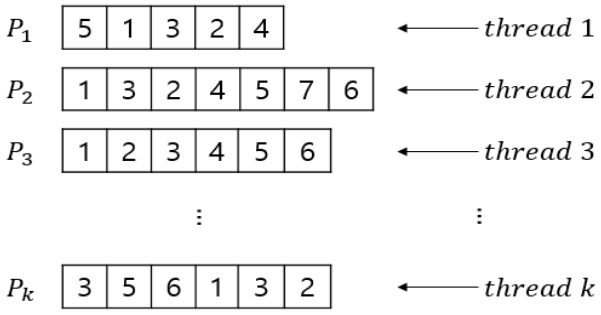
이동테이블 $SHIFT$는 패턴집합 P'이 임의의 q-그램 x에 대해 오문자규칙으로 이동할 수 있는 거리를 저장한다. 이동테이블의 저장된 값을 이용하면, T의 부분문자열 $T[i-m+1...i]$와 순위동형 가능성이 있는 후보패턴 P_i가 존재하는지 판단할 수 있다. 후보패턴 P_i가 존재할 경우($SHIFT[\alpha]=0$인 경우), 해시테이블을 이용하여 후보패턴 P_i를 찾아 $T[i-m+1...i]$와 순위동형인지 접두사테이블을 이용하여 검증하고, 후보패턴 P_i가 없을 경우($SHIFT[\alpha]\neq 0$인 경우), $SHIFT[\alpha]$만큼 P'를 이동시킨다. 이동테이블은 다음의 수식 (1)과 (2)를 이용하여 계산할 수 있다.

$$l=\max\{j:f(P_i[j-q+1..j])=\alpha,1\le i\le k,q\le j\le m\}\ (1)$$

$$SHIFT[\alpha]=\min(m-l,m-q+1)\ (2)$$

먼저 이동테이블을 이동할 수 있는 최대거리인 $m-q+1$로 초기화를 한다. 이후 단계 i에서 P_i에 대해 스텝 $j(q\le j\le m)$를 통해 마지막 위치가 j인 q-그램의 $f(P_i[j-q+1...j])$를 계산하고, 이를 이용하여 이동테이블을 갱신한다. $f(P_i[j-q+1...j])=\alpha$일 때의 j가 l이 된다. 실제 이동거리인 $m-l$이 이미 저장되어 있는 $SHIFT[\alpha]$보다 작다면, $SHIFT[\alpha]$는 $m-l$로 갱신된다. 즉, α가 P'의 가장 오른쪽에서 발생한 위치를 기준으로 실제 이동거리를 계산하여 $SHIFT[\alpha]$에 저장한다(그림 1 참조).

이동테이블은 모든 발생 가능한 q-그램에 대해 이동 정보를 저장하므로 $O(q!)$ 공간이 필요하며, 초기화는 $O(q!)$ 시간에 초기화할 수 있다. 하나의 q-그램에 대한 핑거프린트함수값은 $O(q\log q)$ 시간에 계산될 수 있다. 각 패턴마다 $m-q+1$개의 q-그램이 있고, k개의 패턴이 존재하므로 이동테이블은 총 $O(q!+kmq\log q)$ 시간에 계산될 수 있다.



(그림 2) k개의 스레드를 이용한 이동테이블 병렬계산

검색단계에서는 위에서 계산된 테이블들을 이용하여 $P_i(1 \le i \le k)$와 T의 부분문자열이 순위동형을 이루는 모든 위치를 탐색한다.

3. 이동테이블 병렬계산 알고리즘

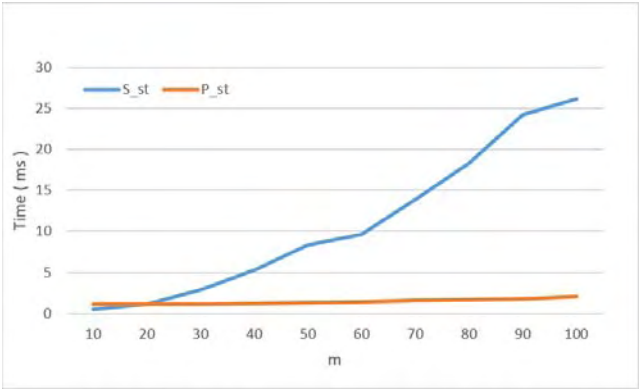
본 논문에서는 k개의 스레드를 사용하여, 패턴집합P'에 대한 이동테이블을 $O(mq\log q)$ 시간에 계산하는 병렬알고리즘을 제시한다. 각 패턴 $P_i(1 \le i \le k)$에 대해 하나의 스레드를 할당하여 후향탐색 방식으로 이동테이블을 계산한다(그림 2 참조). 즉, 각 스레드는 P_i의 q-그램들을 $P_i[m-q+1...m], P_i[m-q...m-1], ..., P_i[1...q]$ 순으로 이동할 수 있는 거리를 계산해 나간다. 예를 들어, 현재 계산한 q-그램의 거리 값과 이동테이블에 저장되어 있는 값을 비교하여 둘 중 더 작은 값을 저장한다. 실제 이동거리가 짧은 q-그램부터 이동테이블을 계산하기 때문에, 스레드간의 경쟁상태(race condition)를 예방할 수 있다. 이와 같은 방법으로 각 스레드는 각 패턴의 $m-q+1$개의 q-그램에 대해서 핑거프린트 값을 계산하면서 패턴집합에 대한 이동테이블을 계산하므로 총 $O(mq\log q)$ 시간에 이동테이블을 계산할 수 있다.

4. 실험 결과

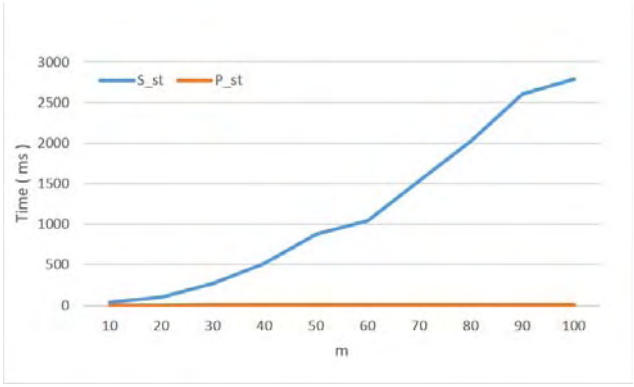
<표 1> 실험 환경

CPU	Intel Core i7-6700
GPU	GeForce GTX1080
OS	Windows 10(64bit)
개발 툴	Visual Studio 2015
개발 언어	C++, CUDA(8.0)

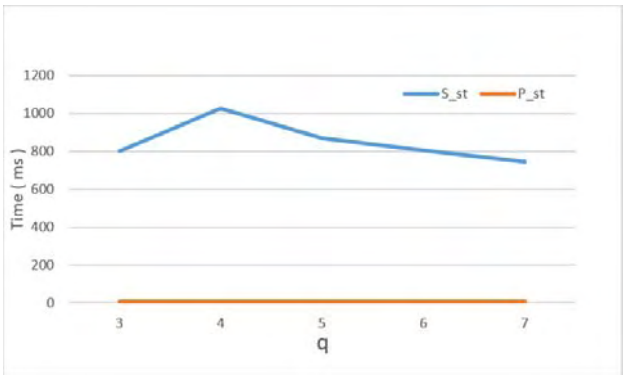
실험 환경은 위의 표 1과 같다. 입력데이터인 패턴집합 P'은 다음과 같이 생성하였다. 입력데이터의 범위는 음악 MIDI(musical instrument digital interface)데이터와 동일하게 101로 설정하였다. 패턴의 개수 k는 100부터 1,000까지 100씩 증가시키며, 패턴의 길이 m은 10부터 100까지 10씩 증가시키며, 패턴들을 무작위로 생성하였다. 각 실험에 대해서 병렬알고리즘은 k개의 스레드블록을 사용하였



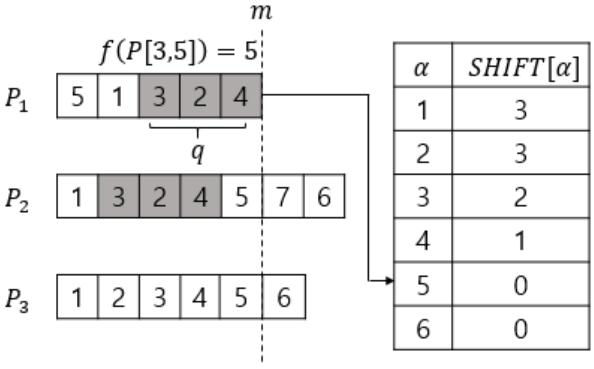
(그림 3) $k=100$, $q=5$일 때 $m(10 \le m \le 100)$에 따른 순차알고리즘과 병렬알고리즘의 이동테이블 계산시간 비교



(그림 4) $k=1{,}000$, $q=5$일 때 $m(10 \le m \le 100)$에 따른 순차알고리즘과 병렬알고리즘의 이동테이블 계산시간 비교



(그림 5) $k=1{,}000$, $m=50$일 때 $q(3 \le q \le 7)$에 따른 순차알고리즘과 병렬알고리즘의 이동테이블 계산시간 비교



(그림 1) $m=5$, $q=3$일 때 이동테이블 계산

고, 스레드블록당 하나의 스레드를 사용하였다. [2]에서 제시된 순차알고리즘을 S_st로, 본 논문에서 제시하는 병렬알고리즘을 P_st로 각각 표기한다. 실험에 측정된 시간은 100번의 실험에 대한 평균수행시간이다. P_st의 수행시간에는 디바이스메모리와 호스트메모리 사이의 데이터를 복사하는 cudaMemcpy() 함수의 수행시간을 포함시켰다.

그림 3은 $k=100$, $q=5$일 때 m의 길이를 변화시키며 S_st와 P_st의 이동테이블을 계산하는 시간을 보여준다. $m=100$일 때 P_st는 S_st보다 약 12.9배 빠르게 수행되었다. 그림 4는 $k=1{,}000$, $q=5$일 때 m의 길이를 변화시키며 S_st와 P_st의 이동테이블을 계산하는 시간을 보여준다. $m=100$일 때 P_st가 S_st보다 약 215배 빠르게 수행되었다. 그림 5는 $k=1{,}000$, $m=50$일 때 q의 크기를 변화시키며 S_st와 P_st의 이동테이블을 계산하는 시간을 보여준다. $q=5$일 때 P_st는 S_st보다 약 135배 빠르게 수행되었다.

참고문헌

[1] J. Kim, P. Eades, R. Fleischer, S. H. Hong, C. S. Iliopoulos, K. Park, S. J. Puglisi, and T. Tokuyama. “Order preserving matching,” Theoretical Computer Science, vol. 525, pp. 68-79, 2014.

[2] M. Han, M. Kang, S. Cho, G. Gu, J. S. Sim, K. Park. “Fast multiple order preserving matching algorithms,” IWOCA, vol. 9538, pp. 248-259, Feb. 2016.

[3] Y. Kim, J. C. Na, J. S. Sim, “Parallel Computation for Extended Edit Distances Using the Shared Memory on GPU,” KIPS Transactions on Computer and Communication Systems, vol. 4, no. 7, pp. 213-218, 2015.

[4] J. Choi, Y. Kim, J. S. Sim “A Parellel Algorithm for the Boxed-Mesh Permutation Pattern Matching Problem,” Proc. of the 43th KIISE Winter Conference, pp. 1372-1374, 2016.

[5] Y. Shin, Y. Kim, J. S. Sim “Parellel Computation of Z-function for Order-Preserving multiple Pattern Matching,” Proc. of the 43th KIISE Winter Conference, pp. 1812-1814, 2016.

[6] R. S. Boyer, J. S. Moore, “A fast string searching algorithm,“ Communications of the ACM, vol. 20, no. 7, 1977.