(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년10월28일
(11) 등록번호 10-1322123
(24) 등록일자 2013년10월18일

(51) 국제특허분류(Int. Cl.)
G06F 17/00 (2006.01) *G06F 17/10* (2006.01)
(21) 출원번호 10-2013-0068361
(22) 출원일자 2013년06월14일
심사청구일자 2013년06월14일
(56) 선행기술조사문헌
JP09006924 A*
KR1020110009098 A
KR1020080102006 A
KR1020120029505 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
인하대학교 산학협력단
인천광역시 남구 인하로 100, 인하대학교 (용현동)
(72) 발명자
심정섭
인천광역시 남동구 구월3동 1476번지 대우재LH 102동 502호
강대웅
울산광역시 동구 화정동 현대파라다이스아파트 702동 701호
김영호
서울특별시 구로구 오류1동 시티월드 701호
(74) 대리인
양성보

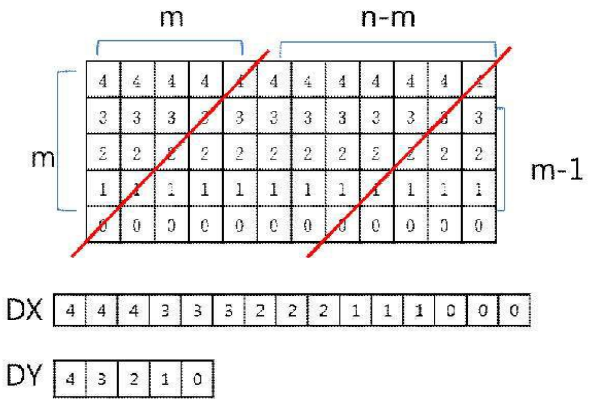
전체 청구항 수 : 총 4 항 심사관 : 이석형

(54) 발명의 명칭 **교환 연산을 포함한 확장편집거리의 계산을 병렬적으로 수행하기 위한 방법**

(57) 요 약

교환 연산을 포함한 확장편집거리의 계산을 병렬적으로 수행하기 위한 방법이 개시된다. 문자열 매칭(character string matching)을 판단하기 위한 거리 함수(distance function) 중 하나인 확장편집거리(extended edit distance)를 계산하는 방법은, 제1 문자열과 제2 문자열에 대하여, 상기 제1 문자열에 포함된 문자와 상기 제2 문자열에 포함된 문자 간의 확장편집거리를 저장하기 위한 테이블을 초기화 하는 초기화 단계; 및 상기 테이블의 각 행 또는 각 열 별로 할당된 쓰레드(thread)를 이용하여 상기 테이블 상에서 대각선 방향에 위치한 셀의 확장편집거리를 병렬적으로 계산하는 계산 단계를 포함한다. 이때, 상기 초기화 단계는, 상기 쓰레드 각각에 대하여 상기 제1 문자열과 상기 제2 문자열 간의 교환(swap) 연산이 발생하는 위치를 저장하기 위한 보조 자료구조를 계산할 수 있고, 상기 계산 단계는, 각 쓰레드 마다 상기 초기화 단계에서 계산된 상기 보조 자료구조를 이용하여 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산할 수 있다.

대 표 도 - 도5



이 발명을 지원한 국가연구개발사업
과제고유번호 1345189451
부처명 교육부
연구사업명 핵심연구(개인)
연구과제명 차세대 유전체서열분석을 위한 근사문자열 알고리즘 개발
주관기관 인하대학교 산학협력단
연구기간 2013.05.01 ~ 2014.04.30
이 발명을 지원한 국가연구개발사업
과제고유번호 1415122511
부처명 지식경제부
연구사업명 디지털콘텐츠원천기술개발
연구과제명 상황대응형 분산 트랜스코딩기술을 이용한 저전력 고성능 멀티미디어 콘텐츠 관리기술 개발
주관기관 인하대학교 산학협력단
연구기간 2013.06.01 ~ 2014.05.31

특허청구의 범위

청구항 1

삭제

청구항 2

문자열 매칭(character string matching)을 판단하기 위한 거리 함수(distance function) 중 하나인 확장편집거리(extended edit distance)를 계산하는 방법에 있어서,

제1 문자열과 제2 문자열에 대하여, 상기 제1 문자열에 포함된 문자와 상기 제2 문자열에 포함된 문자 간의 확장편집거리를 저장하기 위한 테이블을 초기화 하는 초기화 단계; 및

상기 테이블의 각 행 또는 각 열 별로 할당된 쓰레드(thread)를 이용하여 상기 테이블 상에서 대각선 방향에 위치한 셀의 확장편집거리를 병렬적으로 계산하는 계산 단계를 포함하고,

상기 초기화 단계는, 상기 쓰레드 각각에 대하여 상기 제1 문자열과 상기 제2 문자열 간의 교환(swap) 연산이 발생하는 위치를 저장하기 위한 보조 자료구조를 계산하며,

상기 계산 단계는, 각 쓰레드 마다 상기 초기화 단계에서 계산된 상기 보조 자료구조를 이용하여 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하고,

상기 초기화 단계는,

하기 수학식 1을 이용하여 상기 테이블을 초기화 하는 것

을 특징으로 하는 확장편집거리 계산 방법.

수학식 1:

$$\begin{cases} H[0,0] = 0, \\ H[i,0] = i \quad (1 \le i \le m), \\ H[0,j] = j \quad (1 \le j \le n) \end{cases}$$

(여기서, m은 제1 문자열의 길이, n은 제2 문자열의 길이, i는 제1 문자열에 포함된 문자, j는 제2 문자열에 포함된 문자, H[i,0]는 테이블의 첫 번째 행, H[0,j]는 테이블의 첫 번째 열을 나타낸다.)

청구항 3

문자열 매칭(character string matching)을 판단하기 위한 거리 함수(distance function) 중 하나인 확장편집거리(extended edit distance)를 계산하는 방법에 있어서,

제1 문자열과 제2 문자열에 대하여, 상기 제1 문자열에 포함된 문자와 상기 제2 문자열에 포함된 문자 간의 확장편집거리를 저장하기 위한 테이블을 초기화 하는 초기화 단계; 및

상기 테이블의 각 행 또는 각 열 별로 할당된 쓰레드(thread)를 이용하여 상기 테이블 상에서 대각선 방향에 위치한 셀의 확장편집거리를 병렬적으로 계산하는 계산 단계를 포함하고,

상기 초기화 단계는, 상기 쓰레드 각각에 대하여 상기 제1 문자열과 상기 제2 문자열 간의 교환(swap) 연산이 발생하는 위치를 저장하기 위한 보조 자료구조를 계산하며,

상기 계산 단계는, 각 쓰레드 마다 상기 초기화 단계에서 계산된 상기 보조 자료구조를 이용하여 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하고,

상기 보조 자료구조는,

상수 크기의 문자열 ∑로 구성된 상기 제1 문자열과 상기 제2 문자열에 대하여 상기 문자열 ∑에 속하는 각 문자가 상기 제1 문자열에서 마지막으로 발생한 위치를 상기 문자열 ∑의 오름차순으로 저장하기 위한 배열, 및

상기 제1 문자열 중 교환할 문자가 제2 문자열에서 나타나는 마지막 위치를 저장하기 위한 변수로 구성되는 것

을 특징으로 하는 확장편집거리 계산 방법.

청구항 4

문자열 매칭(character string matching)을 판단하기 위한 거리 함수(distance function) 중 하나인 확장편집거리(extended edit distance)를 계산하는 방법에 있어서,

제1 문자열과 제2 문자열에 대하여, 상기 제1 문자열에 포함된 문자와 상기 제2 문자열에 포함된 문자 간의 확장편집거리를 저장하기 위한 테이블을 초기화 하는 초기화 단계; 및

상기 테이블의 각 행 또는 각 열 별로 할당된 쓰레드(thread)를 이용하여 상기 테이블 상에서 대각선 방향에 위치한 셀의 확장편집거리를 병렬적으로 계산하는 계산 단계를 포함하고,

상기 초기화 단계는, 상기 쓰레드 각각에 대하여 상기 제1 문자열과 상기 제2 문자열 간의 교환(swap) 연산이 발생하는 위치를 저장하기 위한 보조 자료구조를 계산하며,

상기 계산 단계는, 각 쓰레드 마다 상기 초기화 단계에서 계산된 상기 보조 자료구조를 이용하여 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하고,

상기 계산 단계는,

하기 수학식 2를 통해 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하는 것

을 특징으로 하는 확장편집거리 계산 방법.

수학식 2:

$$H[i,j] = \min \begin{cases} H[i,j-1]+1 \\ H[i-1,j]+1 \\ H[i-1,j-1]+\delta(X[i],\,Y[j]) \\ s(i,j) \end{cases}$$

(여기서, i는 제1 문자열에 포함된 문자, j는 제2 문자열에 포함된 문자, X[i]는 제1 문자열의 i번째 문자, Y[j]는 제2 문자열의 j번째 문자, H[i,j]는 제1 문자열의 i번째 문자와 제2 문자열의 j번째 문자 간의 확장편집거리를 나타낸다. 그리고, δ(X[i], Y[j])는 X[i]=Y[j]일 때 0이 되고, X[i]≠Y[j]일 때 1이 된다. 또한, s(i,j)는 교환연산에 대한 편집거리를 계산하기 위한 수식으로 $s(i,j) = H[p_{ij}-1, q_{ij}-1] + (i-p_{ij}-1) + (j-q_{ij}-1) + 1$ (이때, $p_{i,j}$와 $q_{i,j}$는 교환연산이 발생하는 위치)와 같이 정의된다.)

청구항 5

문자열 매칭(character string matching)을 판단하기 위한 거리 함수(distance function) 중 하나인 확장편집거리(extended edit distance)를 계산하는 방법에 있어서,

제1 문자열과 제2 문자열에 대하여, 상기 제1 문자열에 포함된 문자와 상기 제2 문자열에 포함된 문자 간의 확장편집거리를 저장하기 위한 테이블을 초기화 하는 초기화 단계; 및

상기 테이블의 각 행 또는 각 열 별로 할당된 쓰레드(thread)를 이용하여 상기 테이블 상에서 대각선 방향에 위치한 셀의 확장편집거리를 병렬적으로 계산하는 계산 단계를 포함하고,

상기 초기화 단계는, 상기 쓰레드 각각에 대하여 상기 제1 문자열과 상기 제2 문자열 간의 교환(swap) 연산이 발생하는 위치를 저장하기 위한 보조 자료구조를 계산하며,

상기 계산 단계는, 각 쓰레드 마다 상기 초기화 단계에서 계산된 상기 보조 자료구조를 이용하여 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하고,

상기 계산 단계는,

상기 제1 문자열의 길이가 m이고 상기 제2 문자열의 길이가 n일 때, 사이클 t가 1≤t≤m인 경우 사이클 마다 계산하는 쓰레드를 하나씩 늘려가면서 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하고, m+1≤t≤n인 경우 매 사이클 마다 계산하는 쓰레드의 수를 유지한 상태로 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하며, n+1≤t≤m+n인 경우 매 사이클 마다 계산하는 쓰레드의 수를 하나씩 줄이면서 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하는 것

을 특징으로 하는 확장편집거리 계산 방법.

명 세 서

기 술 분 야

[0001] 본 발명의 실시예들은 두 문자열 간에 확장편집거리를 계산하기 위한 방법에 관한 것이다.

배 경 기 술

[0002] 본 발명의 배경이 되는 기술은 다음의 논문에 개시되어 있다.

[0003] [1] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer", in Proc.IEEE Symp. Res. Security Privacy, pp.202-212, 1994.

[0004] [2] Baeza-Yates, Ricardo, and Gonzalo Navarro. "Fast approximate string matching in a dictionary." String Processing and Information Retrieval: A South American Symposium, 1998. Proceedings. IEEE, 1998.

[0005] [3] T. F. Smith and M. S. Waterman "Identification of Common Molecular Subsequences", J. Mol. Biol. 147, pp.195-197, 1981.

[0006] 문자열의 불일치를 허용하는 근사문자열매칭(approximate character string matching) 문제는 컴퓨터보안, 검색엔진, 생물정보학 등 많은 분야에서 활발히 연구되고 있다.

[0007] 상기한 논문 중 [1]에서는 네트워크에서 들어오는 패킷에서 일반 데이터와 악성코드를 빠르게 구별하기 위하여 근사문자열매칭 알고리즘을 이용한 침입탐지시스템(intrusion detection system)이 개시되어 있다. 그리고, [2]에서는 사전과 같은 방대한 텍스트 기반 데이터베이스에서 오차를 허용하는 패턴을 실시간으로 찾기 위하여 근사문자열매칭 알고리즘을 이용한 사전 검색 알고리즘이 개시되어 있다. 또한, [3]에서는 두 문자열이 주어졌을 때 두 문자열의 부분문자열 사이에서 가장 유사도가 높은 부분문자열을 찾아내는 지역배치문제를 해결하는 알고리즘이 개시되어 있다. 근사문자열매칭 알고리즘은 생물정보학에서 두 유전체 사이의 유사도를 분석할 때 활용될 수 있다.

[0008] 근사문자열매칭 문제에서 불일치를 판단하는 척도로는 거리함수(distance function)를 사용한다. 대표적인 거리함수는 해밍거리(Hamming distance), 편집거리(edit distance), 확장편집거리(extended edit distance) 등이 있다.

[0009] 길이가 각각 m, n이고 상수 크기의 알파벳 ∑로 구성된 두 문자열 X와 Y가 주어지고, 두 문자열 X와 Y의 해밍거리는 m=n일 때, 두 문자열의 같은 위치에서 서로 다른 문자의 수로 정의된다.

[0010] X와 Y의 편집거리는 X를 Y로 변환하기 위해 필요한 최소 편집연산(edit operation)의 수로 정의된다. 이때 편집연산은 삽입(insertion), 삭제(deletion), 교체(change) 연산으로 구성된다. 도 1을 참조하면, 삽입연산은 문자열의 임의의 위치에 한 문자를 삽입하는 연산이며, 삭제연산은 문자열 내의 임의의 한 문자를 삭제하는 연산이다. 교체연산은 문자열 내의 한 개의 문자를 다른 문자로 교체하는 연산을 뜻한다. 두 문자열 X와 Y의 편집거리는 잘 알려진 동적프로그래밍(dynamic programming)을 이용하여 O(mn) 시간과 공간을 이용하여 계산할 수 있다.

[0011] 편집거리의 문제는 유전자 연구 분야에서 인간 전체 유전자의 유전 형질이 실제 인체에서 어떤 기능을 담당하는지, 어디에 영향을 미치는지 등을 분석하는데 적용되고 있다. 예를 들어, 암 환자의 유전자에서 특정 유전형질이 암을 유발한다고 짐작할 때 이 유전형질을 암 환자 집단의 전체 유전자와 비교해보고 같은 유전형질이 유전자 상의 동일한 위치에 존재한다면 이 유전형질이 암을 유발하는 유전형질이라 짐작할 수 있을 것이다.

[0012] 특정 유전형질이 유전자 상의 어디에 위치하는지를 확인하기 위해 전체 유전자와 일부 유전형질을 비교하게 되는데, 이때 편집거리문제를 해결하는 알고리즘이 적용될 수 있다. 이를 위해 기존 레퍼런스 매핑 툴(reference mapping tool)에서는 교체연산만을 적용해 유전자 비교를 진행하게 되는데 삽입 연산과 삭제 연산을 포함한 계산법이 더 많은 유전정보를 정확히 찾아주며 실제로 레퍼런스 매핑 툴은 편집거리 계산 알고리즘을 적용하고 있다.

[0013] 그리고, 최근 더 많은 유전자 정보를 찾아내기 위해서 교환연산(swap)에 대한 연구가 진행되고 있다. 확장편집거리는 편집거리의 편집연산에 교환연산을 추가한 거리함수이다. 교환연산은 도 1의 (d)에 도시한 바와 같이 한 문자열 내에 존재하는 인접한 두 문자의 위치를 바꾸는 연산을 뜻한다.

[0014] 유전자를 분석하는 분야에서는 인간 유전자를 더 많이 찾아내는 것도 중요하지만 처리 속도 또한 중요한 문제로 인식되고 있다. 최근 GPU(그래픽 처리 유닛)의 성능이 향상됨에 따라 순차 알고리즘들에 대한 병렬화 연구가 활발해지고 있다. 문자열 관련 문제들에 대한 계산 알고리즘 역시 GPU를 활용하여 병렬화한 연구 결과가 제시되고 있다. 실제로 GPU 기반의 프로그래밍 기술인 CUDA(Common Unified Device Architecture)를 이용하여 BarraCUDA, SOAP3 같은 레퍼런스 매핑 툴을 만들어 내고 있다.

[0015] 따라서, GPU를 이용하여 교환연산을 포함한 확장편집거리를 병렬적으로 빠르게 계산하는 것이 다음 세대 유전자 연구를 위해 중요한 연구 과제가 되고 있다.

발명의 내용

해결하려는 과제

[0016] 길이가 각각 m, n인 두 문자열의 확장편집거리를 O(m+n) 시간과 O(mn) 공간에 계산하는 병렬 알고리즘을 제공할 수 있다.

과제의 해결 수단

[0017] 본 발명의 실시예에 따르면, 문자열 매칭(character string matching)을 판단하기 위한 거리 함수(distance function) 중 하나인 확장편집거리(extended edit distance)를 계산하는 방법은, 제1 문자열과 제2 문자열에 대하여, 상기 제1 문자열에 포함된 문자와 상기 제2 문자열에 포함된 문자 간의 확장편집거리를 저장하기 위한 테이블을 초기화 하는 초기화 단계; 및 상기 테이블의 각 행 또는 각 열 별로 할당된 쓰레드(thread)를 이용하여 상기 테이블 상에서 대각선 방향에 위치한 셀의 확장편집거리를 병렬적으로 계산하는 계산 단계를 포함한다. 이때, 상기 초기화 단계는, 상기 쓰레드 각각에 대하여 상기 제1 문자열과 상기 제2 문자열 간의 교환(swap) 연산이 발생하는 위치를 저장하기 위한 보조 자료구조를 계산할 수 있고, 상기 계산 단계는, 각 쓰레드 마다 상기 초기화 단계에서 계산된 상기 보조 자료구조를 이용하여 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산할 수 있다.

[0018] 일 측면에 따르면, 상기 초기화 단계는, 하기 수학식 1을 이용하여 상기 테이블을 초기화 할 수 있다.

[0019] 수학식 1:

[0020]
$$\begin{cases} H[0,0] = 0, \\ H[i,0] = i \quad (1 \le i \le m), \\ H[0,j] = j \quad (1 \le j \le n) \end{cases}$$

[0021] (여기서, m은 제1 문자열의 길이, n은 제2 문자열의 길이, i는 제1 문자열에 포함된 문자, j는 제2 문자열에 포함된 문자, H[i,0]는 테이블의 첫 번째 열, H[0,j]는 테이블의 첫 번째 행을 나타낸다.)

[0022] 다른 측면에 따르면, 상기 보조 자료구조는, 상수 크기의 문자열 ∑로 구성된 상기 제1 문자열과 상기 제2 문자열에 대하여 상기 문자열 ∑에 속하는 각 문자가 상기 제1 문자열에서 마지막으로 발생한 위치를 상기 문자열 ∑의 오름차순으로 저장하기 위한 배열, 및 상기 제1 문자열 중 교환할 문자가 제2 문자열에서 나타나는 마지막 위치를 저장하기 위한 변수로 구성될 수 있다.

[0023] 또 다른 측면에 따르면, 상기 계산 단계는, 하기 수학식 2를 통해 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산할 수 있다.

[0024] 수학식 2:

$$H[i,j] = \min \begin{cases} H[i,j-1]+1 \\ H[i-1,j]+1 \\ H[i-1,j-1]+\delta(X[i],\,Y[j]) \\ s(i,j) \end{cases}$$

[0025]

[0026] (여기서, i는 제1 문자열에 포함된 문자, j는 제2 문자열에 포함된 문자, X[i]는 제1 문자열의 i번째 문자, Y[j]는 제2 문자열의 j번째 문자, H[i,j]는 제1 문자열의 i번째 문자와 제2 문자열의 j번째 문자 간의 확장편집거리를 나타낸다. 그리고, δ(X[i], Y[j])는 X[i]=Y[j]일 때 0이 되고, X[i]≠Y[j]일 때 1이 된다. 또한, s(i,j)는 교환연산에 대한 편집거리를 계산하기 위한 수식으로 $s(i,j) = H[p_{ij}-1, q_{ij}-1] + (i-p_{ij}-1) + (j-q_{ij}-1) + 1$ (이때, $p_{i,j}$와 $q_{i,j}$는 교환연산이 발생하는 위치)와 같이 정의된다.)

[0027] 또 다른 측면에 따르면, 상기 계산 단계는, 상기 제1 문자열의 길이가 m이고 상기 제2 문자열의 길이가 n일 때, 사이클 t가 1≤t≤m인 경우 사이클 마다 계산하는 쓰레드를 하나씩 늘려가면서 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하고, m+1≤t≤n인 경우 매 사이클 마다 계산하는 쓰레드의 수를 유지한 상태로 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산하며, n+1≤t≤m+n인 경우 매 사이클 마다 계산하는 쓰레드의 수를 하나씩 줄이면서 상기 테이블의 각 셀에 해당되는 확장편집거리를 계산할 수 있다.

발명의 효과

[0028] 본 발명의 실시예에 따르면, 교환연산을 포함한 확장편집거리를 병렬적으로 계산함으로써 공간 복잡도인 O(mn)를 해치지 않으면서 시간 복잡도를 O(mn)에서 O(m+n)로 개선할 수 있다. 따라서, O(mn)의 공간 복잡도를 유지하면서 시간 복잡도를 기존 O(mn)에 비해 더욱 빠르게 구현할 수 있으며 두 문자열 길이인 m과 n이 길수록 상대적으로 더 나은 성능을 구현할 수 있다.

도면의 간단한 설명

[0029] 도 1은 문자열 X(=abcdeefa)와 Y(=ahceegif)의 확장편집거리에 포함된 4가지 연산인 삽입 연산, 삭제 연산, 교체 연산, 교환 연산의 예시를 나타낸 것이다.

도 2는 문자열 X(=ababca)와 Y(=cbacab)의 편집거리를 계산하기 위해 D-테이블을 도시한 것이다.

도 3은 문자열 X(=ababca)와 Y(=cbacab)의 확장편집거리를 계산하기 위해 이용되는 H-테이블을 도시한 것이다.

도 4는 문자열 X(=ababca)와 Y(=cbacab)의 확장편집거리를 계산하기 위해 쓰레드를 H-테이블 상에 배치하고 한 사이클에 동시에 수행할 셀들을 대각선으로 표시한 도면이다.

도 5는 본 발명의 일 실시예에 있어서, 확장편집거리를 병렬적으로 계산하는 알고리즘을 설명하기 위한 예시 도면이다.

발명을 실시하기 위한 구체적인 내용

[0030] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0031] 본 실시예들은 인간 전체 유전자의 유전 형질이 실제 인체에서 어떤 기능을 담당하는지, 어디에 어떤 영향을 미치는지, 특정 유전형질이 유전자 상의 어디에 위치하는지 등을 분석하기 위한 유전자 분석 시스템에 적용될 수 있다.

[0032] 본 실시예에서는 교환연산을 포함한 확장편집거리 문제를 해결하기 위한 병렬 알고리즘을 제안한다. 특히, 본 실시예에서는 확장편집거리 문제를 해결하기 위해서 비교하고자 하는 두 문자열 X와 Y 길이의 곱 크기에 해당하는 2차원 테이블을 생성하고 모든 테이블의 셀을 계산할 수 있다.

[0033] 본 명세서의 용어를 먼저 정리하면, 길이가 m인 문자열 X가 주어졌을 때 X[i](1≤i≤m)는 i번째 문자를 의미한다. 그리고, 문자열 X'이 X[i]X[i+1]…X[j](1≤i≤j≤m)이면 이를 X[i‥j]로 표기하고 X의 부분문자열(substring)이라 한다. 앞선 X의 부분문자열에서 i가 1인 경우를 X의 접두사(prefix), j가 m인 경우를 X의 접미사(suffix) 라 한다. 또한, 문자열 X의 길이는 |X|로 표현한다. 배열에 저장된 연속적인 데이터를 표기할 때는 중괄호로 나타낸다. 예를 들어, 배열 A에 0번 인덱스에 위치한 값이 1, 1번 인덱스에 위치한 값이 3, 2번 인덱스에 위치한 값이 5라면 A={1,3,5}로 나타낸다. 데이터가 이차원 배열에 저장된 경우 대괄호로 배열의 위치를 표현한다. 예를 들어, 이차원 배열 A에 i행 , j열의 위치는 A[i,j]로 표현한다.

[0034] 먼저, 편집거리 계산 알고리즘을 간단히 설명하면 다음과 같다.

[0035] 길이가 각각 m, n이고 상수 크기의 알파벳 ∑로 구성된 두 문자열 X와 Y가 주어졌을 때, 편집거리는 잘 알려진 동적프로그래밍을 이용해 편집거리를 계산할 수 있다. 이때 계산하는 (m+1)×(n+1) 크기의 테이블을 D-테이블이라 한다(도 2 참조). D-테이블의 한 셀 D[i,j]는 X[1‥i]와 Y[1‥j]의 편집거리를 저장한다. 따라서, D[m,n]은 두 문자열 X, Y의 편집거리이다.

[0036] 편집거리 계산 알고리즘은 D-테이블의 초기화 단계와 계산 단계로 나뉜다.

[0037] 먼저, 초기화 단계에서 수학식 1을 이용하여 테이블의 첫 번째 행과 첫 번째 열을 초기화 할 수 있다.

수학식 1

[0038]
$$\begin{cases} D[0,0] = 0, \\ D[i,0] = i \quad (1 \le i \le m), \\ D[0,j] = j \quad (1 \le j \le n) \end{cases}$$

[0039] 이후, 계산 단계에서는 수학식 2를 통해 각 D[i,j]를 계산할 수 있다.

수학식 2

[0040]
$$D[i,j] = \min \begin{cases} D[i,j-1] + 1 \\ D[i-1,j] + 1 \\ D[i-1,j-1] + \delta(X[i], Y[j]) \end{cases}$$

[0041] δ(X[i], Y[j])는 X[i]=Y[j]일 때 0이 되고, X[i]≠Y[j]일 때 1이 된다. 각 셀은 O(1) 시간에 계산 가능하고 X와 Y의 편집거리를 계산하기 위해서는 (m+1)×(n+1) 크기의 테이블을 모두 계산해야 하므로 O(mn) 시간과 공간이 필요하다.

[0042] 다음으로, 확장편집거리 계산 알고리즘을 간단히 설명하면 다음과 같다.

[0043] 확장편집거리 계산 알고리즘의 기본 아이디어는 편집거리 계산 알고리즘과 동일하게 길이가 각각 m, n이고 상수 크기의 알파벳 ∑로 구성된 두 문자열 X와 Y가 주어졌을 때 동적프로그래밍 기법을 이용하여 확장편집거리를 계산하는 것이다. 이때 확장편집거리에 포함된 교환연산에 대한 편집거리를 계산하기 위해 교환할 문자의 위치를 저장해야 한다.

[0044] 확장편집거리 계산 알고리즘은 편집거리 계산 알고리즘과 마찬가지로 (m+1)×(n+1) 크기의 테이블을 계산하고 이를 H-테이블이라 한다(도 3 참조). H-테이블의 한 셀 H[i,j]는 X[1‥i]와 Y[1‥j]의 확장편집거리를 저장한다. 따라서, H[m,n]은 두 문자열 X, Y의 확장편집거리를 가진다.

[0045] 확장편집거리 계산 알고리즘 역시 H-테이블의 초기화 단계와 계산 단계로 나뉜다.

[0046] 초기화 단계는 D-테이블의 초기화 단계와 동일하게 테이블의 첫 번째 행과 첫 번째 열을 계산한다. 계산 단계는 수학식 3을 통해 각 H[i,j]를 계산한다.

수학식 3

$$H[i,j] = \min \begin{cases} H[i,j-1]+1 \\ H[i-1,j]+1 \\ H[i-1,j-1]+\delta(X[i], Y[j]) \\ s(i,j) \end{cases}$$

[0047]

[0048] 수학식 3에서 δ(X[i], Y[j])는 X[i]=Y[j]일 때 0이 되고, X[i]≠Y[j]일 때 1이 된다. s(i,j)는 교환연산에 대한 편집거리를 계산하기 위한 수식으로 수학식 4와 같다.

수학식 4

$$s(i,j) = H[p_{ij}-1, q_{ij}-1] + (i-p_{ij}-1) + (j-q_{ij}-1) + 1$$

[0049]

[0050] 여기서, $p_{i,j}$와 $q_{i,j}$는 교환연산이 발생하는 위치로 각각 $1 \le i \le m$이고 $1 \le j \le n$일 때 i보다 작고 $X[p_{i,j}]=Y[j]$인 마지막 위치, 그리고 j보다 작고 $X[i]=Y[q_{i,j}]$인 마지막 위치로 정의할 수 있다.

[0051] 교환연산에 대한 편집거리를 계산하기 위해서 $p_{i,j}$와 $q_{i,j}$를 기억해야 하므로 추가적인 자료구조인 DX 배열과 DY 변수를 이용할 수 있다. 이때, DX 배열은 ∑에 속하는 각 문자가 문자열 X[1‥i-1]에서 마지막으로 발생한 위치를 ∑의 오름차순으로 저장하고 DY 변수는 교환할 문자 X[i]가 문자열 Y[1‥j-1]에서 나타나는 마지막 위치를 저장한다. DX 배열과 DY 변수는 X[1‥i]와 Y[1‥j]에 대한 확장편집거리를 계산한 후 X[i]와 Y[j]에 대한 정보를 갱신한다.

[0052] DY가 DX와 달리 변수인 이유는 테이블의 계산 방향이 한 방향으로 일정하기 때문이다. 현재 테이블을 X[i]와 Y[j]에 대하여 계산한다고 하자(도 2 참조). 이와 같이 테이블을 가로 방향으로 계산하게 되면 DX은 X[i]와 Y[j]를 계산할 때 Y[j]가 X[1‥i-1]에서 마지막으로 발생한 위치를 가지고 있어야 한다. 하지만, DY는 X[i]가 Y에서 나타난 위치를 테이블을 계산할 때마다 갱신할 수 있기 때문에 변수만으로 데이터를 관리할 수 있다. 따라서 DX는 |∑| 크기의 배열을 가지고 DY는 하나의 변수를 가진다.

[0053] 교환연산을 상수시간에 계산 가능하므로 각 H[i,j]는 수학식 3에 의하여 O(1) 시간에 계산할 수 있다. X와 Y의 확장편집거리를 계산하기 위해서는 (m+1)×(n+1) 크기의 테이블을 모두 계산해야 하므로 O(mn) 시간과 공간이 필요하다.

[0054] 본 발명에서는 상기한 확장편집거리 계산 알고리즘에 비해 시간 복잡도를 개선하여 보다 빠르게 확장편집거리를 계산할 수 있는 알고리즘을 제안한다.

[0055] 확장편집거리를 계산할 때 데이터 종속성 문제를 해결하기 위해서는 대각선 방향 계산법, 예컨대 테이블의 오른쪽 위에서부터 왼쪽 아래로 향하는 대각선 방향의 셀을 동시에 처리하는 알고리즘을 적용할 수 있다.

[0056] 데이터 종속성 문제는 다음과 같다.

[0057] 예를 들어, H[i,j]를 계산하기 위해서는 H[i,j-1], H[i-1,j-1], H[i-1,j], $H[p_{i,j}-1, q_{i,j}-1]$ ($1 \le p_{i,j} \le i$, $1 \le q_{i,j} \le j$)가 미리 계산되어야 한다. 따라서, 순차 알고리즘처럼 한 열 혹은 한 행을 병렬적으로 계산하는 방법은 적절하지 않다.

[0058] 두 번째, 테이블을 계산할 때 각 테이블 상의 셀에 하나의 쓰레드를 부여해서 계산하게 되는데 병렬 알고리즘은 순차알고리즘과 달리 쓰레드 마다 DX와 DY를 가진다. DX 배열과 DY 변수는 교환연산이 발생하는 문자의 위치를 저장하는 추가적인 자료 구조이다. 도 4에서 각 셀에 써진 숫자는 쓰레드 번호를 의미한다고 할 때, 각 쓰레드

가 계산해야 하는 X와 Y의 부분문자열이 각각 달라지다. 예를 들어, 도 4에서 쓰레드 0번의 DX는 X[1‥6]의 정보를 가지고 테이블을 계산한다. 하지만 쓰레드 1번의 DX는 X[1‥5]의 정보를 가지고 테이블을 계산해야 한다. 만약 순차 알고리즘처럼 하나의 DX만 유지하게 되면 각 쓰레드가 필요한 정보를 유지할 수 없게 된다.

[0059] 마지막으로, 각 쓰레드마다 DY를 변수로 유지하기 위해서 테이블에 쓰레드를 쓰레드의 번호의 내림차순으로 배치하여 계산한다. 앞서 언급한 바와 같이 확장편집거리 계산 알고리즘에서 DY가 변수인 이유는 테이블의 계산 방향으로 일정하기 때문이다. 쓰레드를 내림차순으로 배치하면 각 쓰레드는 도 4에 도시한 바와 같이 테이블의 한 열을 순차적으로 계산하게 된다. 이렇게 쓰레드의 계산 방향이 한 방향으로 유지되므로 DY 는 변수로 관리할 수 있다.

[0060] 본 실시예에 따른 교환연산을 포함한 확장편집거리 계산 알고리즘은 H-테이블의 초기화 단계와 계산 단계로 나뉜다.

[0061] 초기화 단계에서는 CPU 상에서 수학식 5를 이용하여 H-테이블을 초기화 하고 DX 배열의 전처리를 진행할 수 있다.

수학식 5

[0062]
$$\begin{cases} H[0,0] = 0, \\ H[i,0] = i \quad (1 \le i \le m), \\ H[0,j] = j \quad (1 \le j \le n) \end{cases}$$

[0063] 그러나, 순차알고리즘은 테이블의 한 열을 계산한 이후 다음 열을 계산하기 전에 DX 배열의 정보를 갱신할 수 있지만 병렬 알고리즘의 각 쓰레드는 각각 한 열만 계산하므로 미리 각 쓰레드의 DX 배열을 모두 계산해 두어야 한다.

[0064] 따라서, 본 실시예의 초기화 단계에서는 다음과 같이 DX 배열을 계산한다.

[0065] 처음에 min(m,n)번째 쓰레드가 가지는 DX 배열의 정보는 오직 X[1]에 대한 정보 하나뿐이다. 따라서, min(m,n)번째 쓰레드가 사용할 |∑| 크기의 DX 배열을 선언한 이후 배열의 X[1]과 대응되는 위치에 정보를 기록한다. min(m,n)-1번째 쓰레드가 가지는 DX 배열은 앞선 min(m,n)번째 DX 배열의 내용을 모두 복사한 상태에서 X[2]와 대응되는 위치에 2를 기록한다.

[0066] 이러한 방식의 계산을 반복하면 O(|∑|×min(m,n)) 만큼의 공간을 이용하여 모든 쓰레드의 DX 배열을 O(m) 시간에 계산할 수 있다. 그리고, DY는 각 쓰레드마다 변수의 형태로 테이블을 관리할 수 있다. 따라서, 본 실시예에 따른 초기화 단계의 시간 복잡도는 O(m)이 된다.

[0067] 이후 계산 단계에서는, 두 문자열 X와 Y의 길이를 각각 m, n(m≤n)이라 할 때, 각 셀을 하나의 쓰레드가 수학식 3과 4를 이용하여 계산할 수 있다. 계산 후 DY 변수의 값을 갱신한 후 다음 셀을 계산한다.

[0068] 도 5는 |∑|=3, m=5, n=12일 때 H-테이블과 각 쓰레드의 DX 배열 및 DY 변수를 연속된 메모리 영역으로 묶어 나타내고 있다. 각 블록의 번호는 할당된 쓰레드 번호를 의미한다. H-테이블에서 대각선 영역의 셀은 데이터 종속성 문제를 피할 수 있기 때문에 쓰레드 간 동기화가 가능하므로 도 5와 같이 쓰레드를 할당할 수 있다.

[0069] 본 실시예의 교환연산을 포함한 확장편집거리 계산 알고리즘은 한 사이클 동안 m개의 쓰레드를 이용하여 H-테이블 상에서 대각선 방향에 위치한 셀을 병렬적으로 계산할 수 있다. 이때, 사이클 t에 따라 계산 단계에서 3가지 경우가 발생한다.

[0070] (1) 1≤t≤m: 매 사이클 마다 계산하는 쓰레드를 하나씩 늘려가며 H-테이블을 병렬적으로 계산한다.

[0071] (2) m+1≤t≤n: 매 사이클 마다 계산하는 쓰레드의 수를 유지하며 H-테이블을 병렬적으로 계산한다.

[0072] (3) n+1≤t≤m+n: 매 사이클 마다 계산하는 쓰레드의 수를 하나씩 줄이며 H-테이블을 병렬적으로 계산한다.

[0073] 위 3가지 경우로 보아 알 수 있듯이 본 실시예에서는 H-테이블을 m+n+1 사이클만큼 계산 단계를 수행할 수 있다. 각 사이클은 상수시간이 필요하기 때문에 본 실시예의 교환연산을 포함한 확장편집거리 계산 알고리즘은 O(m+n) 시간과 O(mn) 공간에 수행될 수 있다.

[0074] 본 실시예에서는 교환연산을 포함한 확장편집거리를 계산하는 알고리즘을 병렬화 하여 CUDA로 구현할 수 있다.

[0075] 이와 같이, 본 발명의 실시예에 따르면, 교환연산을 포함한 확장편집거리를 병렬적으로 계산함으로써 공간 복잡도인 O(mn)를 해치지 않으면서 시간 복잡도를 O(mn)에서 O(m+n)로 개선할 수 있다. 따라서, O(mn)의 공간 복잡도를 유지하면서 시간 복잡도를 기존 O(mn)에 비해 더욱 빠르게 구현할 수 있으며 두 문자열 길이인 m과 n이 길수록 상대적으로 더 나은 성능을 구현할 수 있다.

[0076] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPA(field programmable array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0077] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

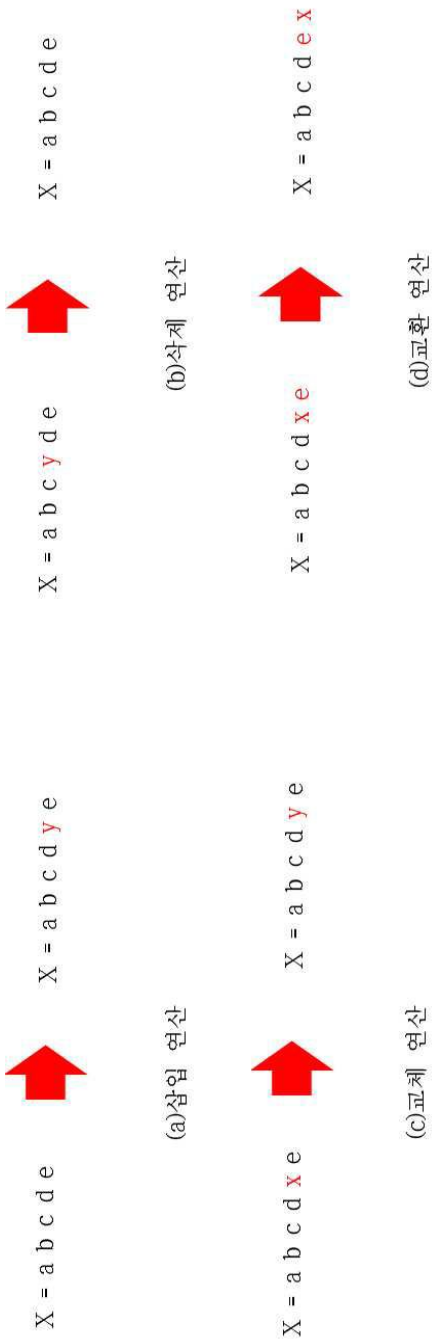
[0078] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0079] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0080] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

도면1


X = a b c d e
X = a b c d y e
(a)삽입 연산
X = a b c d e
X = a b c y d e
(b)삭제 연산
X = a b c d x e
X = a b c d y e
(c)교체 연산
X = a b c d e
X = a b c d e x
(d)교환 연산

도면2

	△	c	b	a	c	a	b
△	0	1	2	3	4	5	6
a	1	1	2	2	3	4	5
b	2	2	1	2	3	4	4
a	3	3	2	1	2	3	4
b	4	4	3	2	2	3	3
c	5	4	4	3	2	3	4
a	6	5	5	4	3	2	3

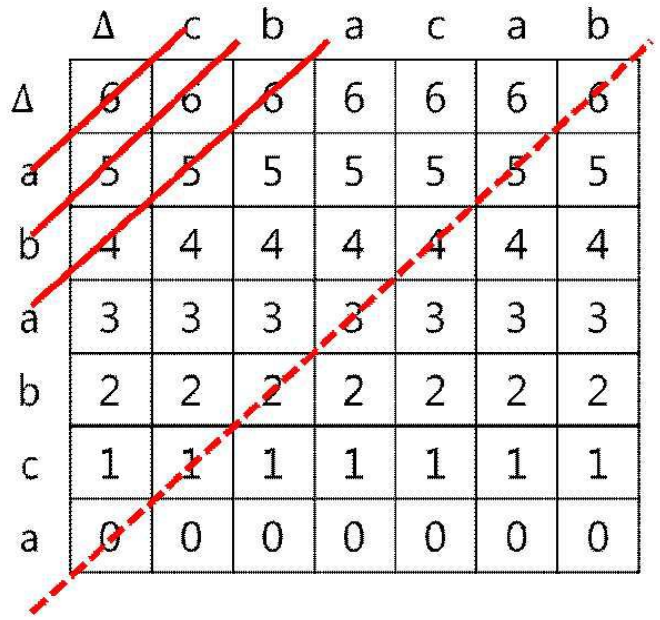
도면3

	Δ	c	b	a	c	a	b
Δ	0	1	2	3	4	5	6
a	1	1	2	2	3	4	5
b	2	2	1	2	3	4	4
a	3	3	2	1	2	3	4
b	4	4	3	2	2	3	3
c	5	4	4	3	2	3	3
a	6	5	5	4	3	2	3

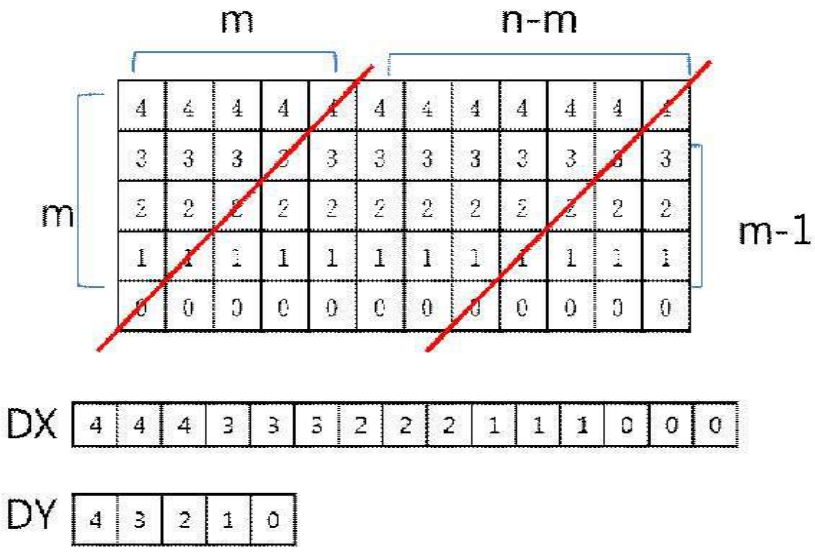
	a	b	c
DX	3	4	5

DY	5

도면4


Δ c b a c a b
Δ 6 6 6 6 6 6 6
a 5 5 5 5 5 5 5
b 4 4 4 4 4 4 4
a 3 3 3 3 3 3 3
b 2 2 2 2 2 2 2
c 1 1 1 1 1 1 1
a 0 0 0 0 0 0 0

도면5


m
n-m
m
m-1
DX 4 4 4 3 3 3 2 2 2 1 1 1 0 0 0
DY 4 3 2 1 0