(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2015년08월11일
(11) 등록번호 10-1543841
(24) 등록일자 2015년08월05일

(51) 국제특허분류(Int. Cl.)
G06F 17/00 (2006.01)
(21) 출원번호 10-2013-0084135
(22) 출원일자 2013년07월17일
심사청구일자 2013년07월17일
(65) 공개번호 10-2015-0009782
(43) 공개일자 2015년01월27일
(56) 선행기술조사문헌
US20100269024 A1
US20120136846 A1
KR101028470 B1

(73) 특허권자
인하대학교 산학협력단
인천광역시 남구 인하로 100, 인하대학교 (용현동)
(72) 발명자
양대헌
서울 서초구 서초중앙로 200, 13동 1407호 (서초동, 삼풍아파트)
(74) 대리인
양성보

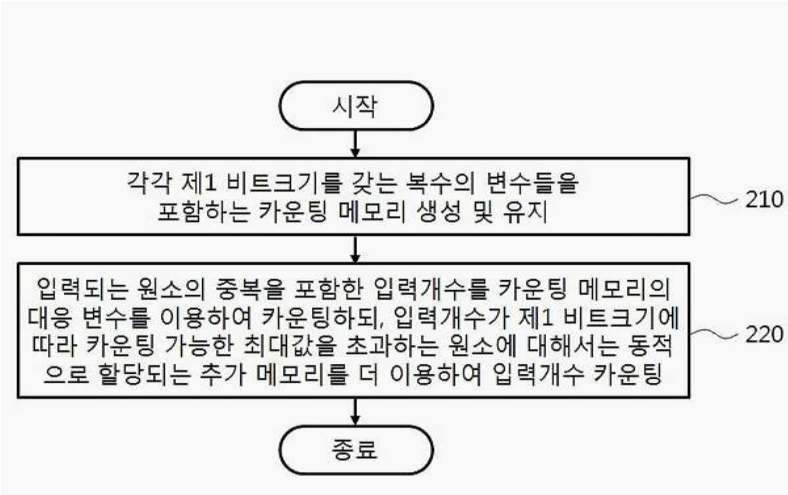
전체 청구항 수 : 총 9 항 심사관 : 이석형

(54) 발명의 명칭 **멀티셋의 개별 원소의 개수를 카운팅하는 방법 및 시스템**

(57) 요 약

멀티셋(multiset)의 개별 원소의 개수를 카운팅하는 방법 및 시스템이 개시된다. 멀티셋의 개별 원소의 개수를 카운팅하기 위해 컴퓨터로 구현되는 카운팅 방법은, 각각 제1 비트크기를 갖는 복수의 변수들을 포함하는 카운팅 메모리를 유지하는 단계 및 입력되는 원소의 중복을 포함한 입력개수를 상기 카운팅 메모리의 대응 변수를 이용하여 카운팅하되, 상기 입력개수가 상기 제1 비트크기에 따라 카운팅 가능한 최대값을 초과하는 원소에 대해서는 동적으로 할당되는 추가 메모리를 더 이용하여 상기 입력개수를 카운팅하는 단계를 포함할 수 있다.

대 표 도 - 도2



이 발명을 지원한 국가연구개발사업
과제고유번호 48276
부처명 교육부
연구관리전문기관 한국연구재단
연구사업명 산학협력 선도대학(LINC) 육성사업
연구과제명 개인 정보 유출 탐지 솔루션 개발
기 여 율 1/1
주관기관 인하대학교 산학협력단
연구기간 2013.03.01 ~ 2014.02.28

명 세 서

청구범위

청구항 1

멀티셋(multiset)의 개별 원소의 개수를 카운팅하기 위해 컴퓨터로 구현되는 카운팅 방법에 있어서,

각각 제1 비트크기를 갖는 복수의 변수들을 포함하는 카운팅 메모리를 유지하는 단계; 및

입력되는 원소의 중복을 포함한 입력개수를 상기 카운팅 메모리의 대응 변수를 이용하여 카운팅하되, 상기 입력개수가 상기 제1 비트크기에 따라 카운팅 가능한 최대값을 초과하는 원소에 대해서는 동적으로 할당되는 추가 메모리를 더 이용하여 상기 입력개수를 카운팅하는 단계

를 포함하는 카운팅 방법.

청구항 2

제1항에 있어서,

상기 대응 변수는, 상기 입력되는 원소를 변수로 갖는 해쉬함수를 이용하여 결정되는 것을 특징으로 하는 카운팅 방법.

청구항 3

제1항 또는 제2항에 있어서,

상기 추가 메모리는, 상기 초과하는 원소의 식별자와 추가변수가 서로 대응되도록 함께 저장되는 확장테이블을 포함하고,

상기 추가변수는 상기 식별자에 대응하는 원소의 추가 카운팅을 위한 제2 비트크기를 갖는 것을 특징으로 하는 카운팅 방법.

청구항 4

제3항에 있어서,

상기 입력개수를 카운팅하는 단계는,

상기 입력되는 원소에 대응하는 상기 카운팅 메모리의 변수의 값이 상기 최대값을 초과하는 경우, 상기 추가변수에 상기 최대값을 누적하고, 상기 카운팅 메모리의 변수의 값을 1로 설정하는 것을 특징으로 하는 카운팅 방법.

청구항 5

제3항에 있어서,

상기 입력개수를 카운팅하는 단계는,

상기 입력되는 원소에 대응하는 상기 카운팅 메모리의 변수의 값이 상기 최대값을 초과하는 경우, 상기 추가변수에 상기 최대값과 노이즈값의 차를 누적하고, 상기 카운팅 메모리의 변수의 값을 상기 노이즈값으로 설정하고,

상기 노이즈값은, 상기 멀티셋에 포함된 모든 원소의 개수의 상기 카운팅 메모리의 변수의 개수에 대한 비율을 포함하는 것을 특징으로 하는 카운팅 방법.

청구항 6

삭제

청구항 7

제3항에 있어서,

상기 원소의 식별자는, 상기 원소 또는 상기 원소를 인자로 갖는 체크섬 함수(checksum function)의 결과값을 포함하는 것을 특징으로 하는 카운팅 방법.

청구항 8

제3항에 있어서,

요청된 원소를 상기 확장테이블에서 검색하는 단계; 및

상기 요청된 원소가 상기 확장테이블에 존재하는 경우, 상기 카운팅 메모리의 변수에 카운팅된 원소의 개수 및 상기 추가변수에 카운팅된 원소의 개수의 합을 상기 요청된 원소의 개수로 산출하는 단계

를 더 포함하는 카운팅 방법.

청구항 9

제3항에 있어서,

요청된 원소를 상기 확장테이블에서 검색하는 단계; 및

상기 요청된 원소가 상기 확장테이블에 존재하는 경우, 상기 카운팅 메모리의 변수에 카운팅된 원소의 개수와 상기 추가변수에 카운팅된 원소의 개수의 합에서 노이즈값을 뺀 값을 상기 요청된 원소의 개수로 산출하는 단계

를 더 포함하고,

상기 노이즈값은, 상기 멀티셋에 포함된 모든 원소의 개수의 상기 카운팅 메모리의 변수의 개수에 대한 비율을 포함하는 카운팅 방법.

청구항 10

멀티셋(multiset)의 개별 원소의 개수를 카운팅하는 카운팅 시스템에 있어서,

적어도 하나의 스토리지 유닛; 및

적어도 하나의 프로세서

를 포함하고,

상기 적어도 하나의 프로세서는,

각각 제1 비트크기를 갖는 복수의 변수들을 포함하는 카운팅 메모리를 상기 적어도 하나의 스토리지 유닛에 유지하는 과정; 및

입력되는 원소의 중복을 포함한 입력개수를 상기 카운팅 메모리의 대응 변수를 이용하여 카운팅하되, 상기 입력개수가 상기 제1 비트크기에 따라 카운팅 가능한 최대값을 초과하는 원소에 대해서는 상기 적어도 하나의 스토리지 유닛에 동적으로 할당되는 추가 메모리를 더 이용하여 상기 입력개수를 카운팅하는 과정

을 처리하는 것을 특징으로 하는 카운팅 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명의 실시예들은 멀티셋의 개별 원소의 개수를 카운팅하는 방법 및 시스템에 관한 것이다.

배 경 기 술

[0002] 멀티셋(multiset)은 집합의 일종이나, 집합(set)과는 다르게 같은 원소를 여러 번 저장할 수 있는 집합이다. 예를 들어, {1, 2, 3, 4, 5, 6, 7}은 집합을, {1, 2, 2, 2, 3, 3, 4, 5, 6, 7, 7}은 멀티셋을 각각 나타낼 수 있다. 이러한 멀티셋은, 새로운 원소를 추가할 수도 있고, 멀티셋의 특정 원소가 몇 번 저장되어있는지도 알아낼 수 있다. 예를 들어, 상술한 멀티셋 {1, 2, 2, 2, 3, 3, 4, 5, 6, 7, 7}에서 원소 2의 개수는 3이고, 원소

6의 개수는 1이다.

[0003] 어떤 원소가 멀티셋에 존재하는지를 확인하고, 존재한다면 그 원소가 몇 번 멀티셋에 저장되어있는지를 알아내는 것은 여러 분야에서 매우 중요한 문제이다. 예를 들어, 네트워크에서 하나의 출발지 IP주소에서 몇 개의 패킷을 보냈는지를 확인하기 위해 사용할 수도 있고, 사회관계망에서 발생된 수많은 단어들 중 특정 단어가 몇 번 등장했는지 알아내기 위해서도 사용할 수 있다.

[0004] 가장 쉽게 이를 구현할 수 있는 자료 구조는 테이블이다. 예를 들어, 어떤 단어가 새롭게 입력될 때마다, 그 단어를 카운터와 함께 테이블에 저장해 놓는 것이다. 새로운 단어라면 테이블에 새로운 엔트리를 추가하고 카운터는 1로 설정한다. 만약, 이미 멀티셋에 존재하는 단어라면 해당 카운터 값을 하나 증가시키면 된다. 특정 단어가 멀티셋에 몇 번 등장하는지를 알고 싶다면, 테이블을 검색해서 카운터 값을 돌려주면 된다.

[0005] 그러나, 최근에는 인터넷을 통해 유통되는 데이터의 양이 너무 많아져서(Big data 처리), 이런 단순한 셈 방법으로 다루기에는 너무 많은 양의 메모리가 요구되는 문제점이 있다.

[0006] 일반적으로, 네트워크에서의 소스 IP나 검색분야에서의 키워드들은 그 종류수가 매우 많은 반면, 중복입력이 매우 잦은 소스 IP나 키워드들은 그 수가 그리 많지 않다. 예를 들어, 서버로 전송되는 패킷들은 대부분 몇몇의 소스 IP들에 의해 전송되지, 전체 소스 IP들에게서 골고루 전송되지는 않는다. 뿐만 아니라, 키워드들 역시 자주 쓰이는 몇몇의 키워드들(일례로, "여행", "맛집")이 반복적으로 중복 입력되는 경우가 많을 뿐, 모든 키워드들이 골고루 수신되지는 않는다. 일례로, 키워드 "신경심리학(neuropsychology)"과 같은 키워드가 중복 입력되는 횟수와 키워드 "맛집"과 같은 키워드가 중복 입력되는 횟수는 큰 차이가 존재한다.

[0007] 그럼에도 불구하고 종래기술에서는 각각의 원소들에 대해 얼마만큼의 카운팅이 필요한가를 알 수 없기 때문에 원소들 각각의 개수를 카운팅하기 위해 필요한 변수의 크기를 모든 원소에 대해 매우 크게(일례로, 가장 많이 입력될 것으로 판단되는 원소의 개수에 맞게) 설정할 수 밖에 없으며, 이에 따라 필연적으로 매우 큰 메모리가 요구된다.

[0008] 도 1은 종래기술에 있어서, 개별 원소의 개수를 카운팅한 결과를 나타낸 도면이다. 도 1에서는 네 개의 키워드 각각의 입력개수를 카운팅하기 위해, 각각 16-비트 크기의 변수를 이용한다고 가정한다. 이 경우, 각각의 변수의 값은 0부터 65,535(2^{16}-1)까지의 값 중 하나를 가질 수 있다. 즉, 각각의 키워드들은 각각의 변수를 이용하여 0부터 65,535(2^{16}-1)까지 입력개수가 카운팅될 수 있다. 이때, 키워드 A, B, C, D가 각각 50번, 25번, 60,000번, 250번 입력되었다고 가정한다. 이미 설명한 바와 같이, 원소들 중 일부의 원소들만이 다수 입력될 뿐, 대부분의 원소들은 다수 입력되는 횟수가 매우 작다.

[0009] 이때, 도 1을 살펴보면, 대부분의 메모리들이 불필요하게 낭비되고 있음을 알 수 있다. 그러나 각각의 원소들이 몇 번 입력될지는 알 수 없기 때문에, 변수의 크기를 원하는 데로 줄일 수도 없는 문제점이 있다. 만약, 8-비트 크기의 변수들을 이용한다면, 255번째(2^8-1) 이후에는 다시 0번째로 카운팅되기 때문에 키워드 C에 대해서는 제대로 카운팅을 할 수가 없다. 멀티셋에서는 많이 입력되는 원소들에 대한 정보가 주로 유용하게 이용되기 때문에 많이 입력되는 원소들 각각의 개수를 카운팅할 수 없다는 것은 매우 큰 문제점 된다.

발명의 내용

해결하려는 과제

[0010] 매우 적은 양의 메모리만으로 멀티셋의 개별 원소의 개수를 카운팅할 수 있는 카운팅 방법 및 시스템을 제공한다.

과제의 해결 수단

[0011] 멀티셋(multiset)의 개별 원소의 개수를 카운팅하기 위해 컴퓨터로 구현되는 카운팅 방법에 있어서, 각각 제1 비트크기를 갖는 복수의 변수들을 포함하는 카운팅 메모리를 유지하는 단계; 및 입력되는 원소의 중복을 포함한 입력개수를 상기 카운팅 메모리의 대응 변수를 이용하여 카운팅하되, 상기 입력개수가 상기 제1 비트크기에 따라 카운팅 가능한 최대값을 초과하는 원소에 대해서는 동적으로 할당되는 추가 메모리를 더 이용하여 상기 입력개수를 카운팅하는 단계를 포함하는 카운팅 방법이 제공된다.

[0012] 일측에 따르면, 상기 대응 변수는, 상기 입력되는 원소를 변수로 갖는 해쉬함수를 이용하여 결정되는 것을 특징

으로 할 수 있다.

[0013] 다른 측면에 따르면, 상기 추가 메모리는, 상기 초과하는 원소의 식별자와 상기 식별자에 대응하는 원소의 추가 카운팅을 위한 제2 비트크기를 갖는 추가변수를 서로 연관하여 저장하는 확장테이블을 포함하는 것을 특징으로 할 수 있다.

[0014] 또 다른 측면에 따르면, 상기 입력개수를 카운팅하는 단계는, 상기 입력되는 원소에 대응하는 상기 카운팅 메모리의 변수의 값이 상기 최대값을 초과하는 경우, 상기 추가변수에 상기 최대값을 누적하고, 상기 카운팅 메모리의 변수의 값을 1로 설정하는 것을 특징으로 할 수 있다.

[0015] 또 다른 측면에 따르면, 상기 입력개수를 카운팅하는 단계는, 상기 입력되는 원소에 대응하는 상기 카운팅 메모리의 변수의 값이 상기 최대값을 초과하는 경우, 상기 추가변수에 상기 최대값과 노이즈값의 차를 누적하고, 상기 카운팅 메모리의 변수의 값을 상기 노이즈값으로 설정하는 것을 특징으로 할 수 있다.

[0016] 또 다른 측면에 따르면, 상기 노이즈값은, 상기 멀티셋에 포함된 모든 원소의 개수의 상기 카운팅 메모리의 변수의 개수에 대한 비율을 포함하는 것을 특징으로 할 수 있다.

[0017] 또 다른 측면에 따르면, 상기 원소의 식별자는, 상기 원소 또는 상기 원소를 인자로 갖는 체크섬 함수(checksum function)의 결과값을 포함하는 것을 특징으로 할 수 있다.

[0018] 또 다른 측면에 따르면, 상기 카운팅 방법은 요청된 원소를 상기 확장테이블에서 검색하는 단계; 및 상기 요청된 원소가 상기 확장테이블에 존재하는 경우, 상기 카운팅 메모리의 변수에 카운팅된 원소의 개수 및 상기 추가변수에 카운팅된 원소의 개수를 이용하여 상기 요청된 원소의 개수를 산출하는 단계를 더 포함할 수 있다.

[0019] 또 다른 측면에 따르면, 상기 카운팅 방법은 요청된 원소를 상기 확장테이블에서 검색하는 단계; 및 상기 요청된 원소가 상기 확장테이블에 존재하는 경우, 상기 카운팅 메모리의 변수에 카운팅된 원소의 개수, 상기 추가변수에 카운팅된 원소의 개수 및 노이즈값을 이용하여 상기 요청된 원소의 개수를 산출하는 단계를 더 포함할 수 있다.

[0020] 멀티셋(multiset)의 개별 원소의 개수를 카운팅하는 카운팅 시스템에 있어서, 적어도 하나의 스토리지 유닛; 및 적어도 하나의 프로세서를 포함하고, 상기 적어도 하나의 프로세서는, 각각 제1 비트크기를 갖는 복수의 변수들을 포함하는 카운팅 메모리를 상기 적어도 하나의 스토리지 유닛에 유지하는 과정; 및 입력되는 원소의 중복을 포함한 입력개수를 상기 카운팅 메모리의 대응 변수를 이용하여 카운팅하되, 상기 입력개수가 상기 제1 비트크기에 따라 카운팅 가능한 최대값을 초과하는 원소에 대해서는 상기 적어도 하나의 스토리지 유닛에 동적으로 할당되는 추가 메모리를 더 이용하여 상기 입력개수를 카운팅하는 과정을 처리하는 것을 특징으로 하는 카운팅 시스템이 제공된다.

발명의 효과

[0021] 매우 적은 양의 메모리만으로 멀티셋의 개별 원소의 개수를 카운팅할 수 있다.

도면의 간단한 설명

[0022] 도 1은 종래기술에 있어서, 개별 원소의 개수를 카운팅한 결과를 나타낸 도면이다.

도 2는 본 발명의 일실시예에 있어서, 카운팅 방법을 도시한 흐름도이다.

도 3은 본 발명의 일실시예에 있어서, 개별 원소의 개수를 카운팅한 결과를 나타낸 도면이다.

도 4 내지 도 6은 종래기술에 따른 카운팅 방법의 성능과 본 발명의 일실시예에 따른 카운팅 방법의 성능을 비교하기 위한 도면들이다.

발명을 실시하기 위한 구체적인 내용

[0023] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0024] 도 2는 본 발명의 일실시예에 있어서, 카운팅 방법을 도시한 흐름도이다. 본 실시예에 따른 카운팅 방법은 적어도 하나의 스토리지 유닛과 적어도 하나의 프로세서를 포함하는 컴퓨터로 구현되는 카운팅 시스템 또는 상기 적어도 하나의 프로세서에 의해 수행될 수 있다. 도 2에서는 카운팅 시스템이 카운팅 방법에 포함된 단계들을 수행하는 실시예를 설명한다.

[0025] 단계(210)에서 카운팅 시스템은 각각 제1 비트크기를 갖는 복수의 변수들을 포함하는 카운팅 메모리를 생성 및 유지할 수 있다. 일례로, 카운팅 메모리는 상술한 적어도 하나의 스토리지 유닛에 저장될 수 있다. 카운팅 메모리는 기설정된 개수(일례로, cn개, 변수 cn은 자연수)의 변수를 가질 수 있으며, 각각의 변수들은 제1 비트크기(일례로, c-비트)를 가질 수 있다. 예를 들어, 카운팅 메모리는 'A[cn]'으로 표현되는 배열의 자료구조를 갖도록 구현될 수 있으며, 카운팅 메모리의 i-번째 변수는 c-비트의 크기를 갖는 변수 'A[i]'와 같이 표현될 수 있다. 이때, 각각의 변수들은 0부터 (2^c-1)까지의 값을 가질 수 있다. 다시 말해, 카운팅 시스템은 카운팅 메모리를 이용하여 개별 원소의 개수를 0부터 (2^c-1)까지 카운팅할 수 있다.

[0026] 단계(220)에서 카운팅 시스템은 입력되는 원소의 중복을 포함한 입력개수를 카운팅 메모리의 대응 변수를 이용하여 카운팅하되, 입력개수가 제1 비트크기에 따라 카운팅 가능한 최대값을 초과하는 원소에 대해서는 동적으로 할당되는 추가 메모리를 더 이용하여 입력개수를 카운팅할 수 있다.

[0027] 예를 들어, 멀티셋이 네트워크에서 패킷이 소스(source) IP마다 몇 번 수신되었는가를 판단하기 위해 이용되는 경우, 원소의 입력은 패킷의 수신에 대응될 수 있다. 이때, 하나의 원소에 대해 카운팅되는 입력개수는 하나의 소스 IP가 몇 개의 패킷을 전송하였는가를 의미할 수 있다.

[0028] 다른 예로, 멀티셋이 검색분야에서 인기 키워드를 판단하기 위해 이용되는 경우, 원소의 입력은 키워드의 입력에 대응될 수 있다. 이때, 하나의 원소에 대해 카운팅되는 입력개수는 하나의 키워드가 몇 번 입력되었는가를 의미할 수 있다.

[0029] 카운팅 시스템은 단계(220)에서 카운팅 메모리의 변수들 중 입력되는 원소에 대응하는 변수를 찾고, 대응하는 변수를 이용하여 입력되는 원소의 입력개수를 카운팅할 수 있다. 예를 들어, 입력되는 원소에 대응하는 변수가 i-번째 변수 'A[i]'의 값이 0이라면, 입력되는 원소는 처음 입력되는 원소일 수 있다. 이 경우, 카운팅 시스템은 i-번째 변수 'A[i]'의 값을 0에서 1로 변경하여 해당 원소가 1번 입력되었음을 카운팅할 수 있다. 다른 예로, i-번째 변수 'A[i]'의 값이 3이라면, 입력되는 원소는 이미 세 번 입력되었고, 네 번째 입력되는 원소이기 때문에, 카운팅 시스템은 i-번째 변수 'A[i]'의 값을 3에서 4로 변경하여 현재 입력되는 원소가 네 번째 입력되는 원소임을 카운팅할 수 있다.

[0030] 이때, i-번째 변수 'A[i]'의 값이 각 변수가 가질 수 있는 최대값 (2^c-1)이라면, 카운팅 시스템은 i-번째 변수 'A[i]'의 값을 더 이상 증가시킬 수 없기 때문에, i-번째 변수 'A[i]'만으로는 입력되는 원소의 중복을 포함하는 입력개수를 더 이상 카운팅할 수 없다. 따라서, 카운팅 시스템은 입력개수가 제1 비트크기에 따라 카운팅 가능한 최대값을 초과하는 원소에 대해서는 동적으로 할당되는 추가 메모리를 더 이용하여 입력개수를 카운팅할 수 있다.

[0031] 따라서, 본 발명의 실시예들에 따르면, 개별 원소의 개수를 일정 크기의 변수를 이용하여 카운팅하되, 해당 변수를 이용하여 카운팅할 수 없는 원소의 개수에 대해서만 추가 메모리를 이용하여 카운팅함으로써, 요구되는 메모리의 크기를 획기적으로 줄일 수 있다.

[0032] 도 3은 본 발명의 일실시예에 있어서, 개별 원소의 개수를 카운팅하기 위해 필요한 메모리를 나타낸 도면이다. 도 3은 카운팅 메모리가 포함하는 변수들 각각의 크기를 8-비트라 가정하고, 도 1의 종래기술에서와 동일하게 키워드 A, B, C, D가 각각 50번, 25번, 60,000번, 250번 입력되었다고 가정한다. 이때, 본 실시예에서는 원소의 입력개수가 8-비트의 크기를 초과하는 경우, 추가변수(310)를 더 이용하여 카운팅함으로써, 요구되는 메모리의 크기를 획기적으로 줄일 수 있다. 도 3에서는 추가변수(310)로 16-비트 크기의 변수가 사용된 예를 설명하고 있다.

[0033] 또한, 도 1 및 도 3에서는 단지 4개의 키워드를 이용한 예를 들었다. 그러나 만약, 10,000개의 키워드가 존재하고, 주로 많이 이용되는 키워드 1,000개만이 255번을 초과하여 입력되며, 9,000개의 키워드들이 모두 255번 이하로 입력된다고 가정하자. 이때, 종래기술에서는 10,000개의 16-비트 크기의 변수들이 필요한 반면, 본 발명의 실시예에서는 1000개의 16-비트 크기의 변수와 9,000개의 8-비트 크기의 변수들이 필요하기 때문에 요구되는 메모리의 크기가 획기적으로 감소함을 알 수 있다. 만약, 16-비트 크기의 변수로도 카운팅할 수 없는 키워드가 존재하여 32-비트 크기($2^{32}-1 = 4294967295$)의 변수를 사용해야만 한다고 가정하면, 본 발명의 실시예를 이용하여 감소시킬 수 있는 메모리의 크기가 기하급수적으로 증가함을 알 수 있다.

[0034] 추가변수(310)를 더 이용하는 방법은 다음과 같다. 입력되는 원소에 대한 카운팅 메모리의 변수가 이미 최대값

이라면, 도 2를 통해 설명한 카운팅 시스템은 추가변수(310)를 할당하고, 추가변수(310)의 값(초기값은 '0')으로 추가변수(310)의 기존값과 최대값의 합을 할당할 수 있다. 또한, 카운팅 시스템은 카운팅 메모리의 변수의 값을 현재 입력되는 원소를 위한 '1'로 설정할 수 있다.

[0035] 예를 들어, 도 3의 예에서 키워드 C가 256번째 입력되는 경우, 카운팅 메모리의 변수가 이미 최대값을 갖고 있다. 따라서, 카운팅 시스템은 추가변수(310)를 할당하고 추가변수(310)의 값을, 추가변수(310)의 기존값과 최대값의 합인 255(0+255)로 설정할 수 있다. 이때, 카운팅 메모리의 변수의 값은 '1'로 설정될 수 있다. 이 경우, 카운팅 메모리의 변수의 값 '1'과 추가변수(310)의 값 255를 통해 키워드 C가 256번 입력되었음을 확인할 수 있다. 키워드 C가 511번째 입력되는 경우에도, 카운팅 메모리의 변수는 다시 최대값을 갖는다. 이 경우, 카운팅 시스템은 추가변수(310)의 값을, 추가변수(310)의 기존값과 최대값의 합인 510(255+255)로 설정할 수 있다. 이때, 카운팅 메모리의 변수의 값은 '1'로 설정될 수 있다. 이 경우에도, 카운팅 메모리의 변수의 값 '1'과 추가변수(310)의 값 510을 통해 키워드 C가 511번 입력되었음을 확인할 수 있다.

[0036] 다른 실시예에서 추가 메모리는, 어떠한 원소의 입력개수가 제1 비트크기에 따라 카운팅 가능한 최대값을 초과하였는지 확인하기 위해, 초과하는 원소의 식별자와 상기 식별자에 대응하는 원소의 추가 카운팅을 위한 제2 비트크기를 갖는 변수를 서로 연관하여 저장하는 확장테이블을 포함할 수 있다. 이때, 원소의 식별자는 해당 원소 자체이거나 또는 해당 원소를 인자로 갖는 체크섬 함수(checksum function)의 결과값인 해당 원소의 체크섬일 수 있다. 예를 들어, 키워드 "여행"의 식별자는 "여행"이거나 또는 "여행"의 체크섬일 수 있다. 또한, 도 3의 추가변수(310)가 본 실시예의 제2 비트크기를 갖는 변수에 대응될 수 있다. 예를 들어, 확장테이블에는 키워드 C나 키워드 C의 체크섬이 추가변수(310)와 서로 연관하여 저장될 수 있다. 확장테이블은 (E[i].kwd, E[i].kwdcnt)와 같이 표현될 수 있다. E[i].kwd는 i-번째 원소 또는 i-번째 원소의 체크섬을 의미할 수 있고, E[i].kwdcnt는 i-번째 원소에 대응하는 제2 비트크기의 변수(이하, 추가변수)를 의미할 수 있다.

[0037] 도 2의 실시예에서는 카운팅 메모리가 배열의 자료구조를 갖는 것으로 설명하였으나, 카운팅 메모리의 변수들과 입력되는 원소들이 1:1로 매칭되는 경우, 어떠한 원소가 어떠한 변수가 매칭되는가를 확인하기 위해 카운팅 메모리 역시 테이블의 형태로 구현될 수도 있다.

[0038] 반면, 또 다른 실시예에서는, 보다 적은 양의 메모리를 이용하여 멀티셋의 개별 원소를 카운팅하기 위해 카운팅 메모리의 변수들과 입력되는 원소들이 1:1로 매칭되지 않을 수도 있다. 이를 위해, 본 실시예에서는 정확한 셈(counting) 대신, 멀티셋을 위한 근사치 셈(approximate counting for multiset)이 이용될 수 있다.

[0039] 이때, 카운팅 메모리는 도 2의 실시예에서와 같이 배열의 자료구조를 이용할 수 있고, 입력되는 원소에 대응하는 카운팅 메모리의 변수는 입력되는 원소를 인자로 갖는 해쉬함수를 이용하여 결정될 수 있다. 해쉬함수 H()는 임의의 길이의 입력 비트열을 고정길이의 랜덤한 비트열로 변환하고, 같은 입력에 대해서는 같은 출력을 내는 함수일 수 있다. 예를 들어, 키워드 A가 입력되면, 카운팅 메모리 A[cn]의 변수 A[h]에 키워드 A의 입력개수가 카운팅될 수 있다. 여기서, h는 해쉬함수 H(키워드 A)의 결과값(해쉬값)일 수 있다. 대부분의 경우, 서로 다른 키워드는 서로 다른 변수를 통해 카운트가 된다. 그러나, 서로 다른 키워드이지만 같은 해쉬값을 갖는 경우가 존재하며, 이 경우에는 에러가 발생할 수 있다. 본 실시예에서는 에러를 통해 발생하는 노이즈를 제거함으로써, 멀티셋을 위한 근사치 셈을 이용할 수 있다.

[0040] 이미 설명한 바와 같이, 추가 메모리의 추가변수는 카운팅 메모리의 변수가 최대값이 될 때마다 최대값이 누적될 수 있다. 그러나, 해쉬함수를 이용하여 카운팅 메모리의 변수와 멀티셋의 원소를 매칭하는 실시예에서는 노이즈가 발생할 수 있기 때문에, 본 실시예에서는 노이즈를 tot/cn와 같이 계산할 수 있다. 여기서 tot는 현재 입력된 중복을 포함한 모든 키워드의 전체개수를 의미할 수 있고, 노이즈 tot/cn는 해당 원소가 아니라, 다른 원소가 해쉬함수를 통해 해당 원소의 카운트를 증가시켰을 것으로 추정되는 값을 의미할 수 있다. 예를 들어, 현재 입력되는 원소가 100만 번째 입력이고, 카운팅 메모리의 변수의 개수인 cn이 만개라면, 노이즈는 100이 될 수 있다. 이때, 추가변수에는 최대값이 아니라 최대값에서 노이즈를 뺀 값이 누적될 수 있다. 예를 들어, 카운팅 메모리의 변수가 최대값일 때, 추가변수의 값은 다음과 같이 알고리즘적으로 표현될 수 있다.

[0041] $E[found].kwdcnt \leftarrow E[found].kwdcnt + (2^c-1) - tot/cn$

[0042] 여기서, 'found'는 입력되는 원소의 대응하는 확장테이블에서의 위치를 의미할 수 있다. 또한, 카운팅 메모리의 변수 역시 '1'로 설정되는 것이 아니라, 'tot/cn'으로 설정될 수 있다. 노이즈 'tot/cn'는 자연수가 아닐 수 있다. 그러나, 본 실시예는 근사치 셈(approximate counting for multiset)을 이용하는 것이기 때문에 자연수가 아닌 변수의 값 및 추가변수의 값이 허용될 수 있다.

[0043] 어떤 원소(키워드)가 주어질 때, 키워드의 개수를 카운팅하기 위한 알고리즘은 다음 표 1과 같이 표현될 수 있다.

표 1

[0044]

```
1. tot ← tot+1와 같이 현재 입력된 중복을 포함한 모든 키워드의 전체개수를 카운팅.
2. 확장테이블에서 키워드(keyword)를 검색.
3. 만약, 이미 keyword가 확장테이블의 i-번째에 존재한다면(즉, E[i].kwd와 keyword 또는 keyword의 체크섬이 같다면),
        found ← i
4. 만약, keyword가 확장테이블에 존재하지 않는다면, 확장테이블에 다음과 같이 새로운 엔트리를 등록(empty번째 자리가 비었다고 가정).
        E[empty].kwd ← keyword
        E[empty].expcnt ← 0
        found ← empty
5. 다음을 수행:
        5.1 current_counter ← H(keyword, E[found].kwdcnt)
        5.2 만약 A[current_counter]이 값이 2^c-1로 최대치라면,
                a. E[found].kwdcnt  ← E[found].kwdcnt + (2^c-1) - tot/cn
                b. A[current_counter] ← tot/cn
                c. 4.1로 다시 반복
        5.3 그렇지 않다면(A[current_counter]이 값이 최대치가 아니라면), A[current_counter]를 1 증가시킴.
```

[0045] 또한, 주어진 원소(키워드)가 몇 번 저장되었는지를 확인하기 위한 알고리즘은 다음 표 2와 같이 표현될 수 있다.

표 2

[0046]

```
1. 키워드(keyword)를 확장테이블 E[]에서 검색.
2. 검색결과, keyword가 존재하지 않는다면, 0을 리턴(이 멀티셋에 keyword가 존재하지 않음).
3. 검색결과 keyword가 확장테이블의 found-번째에 존재한다면, 다음을 계산:
   3.1 current_counter ← H(keyword, E[found].kwdcnt)
   3.2 ec ← E[found].kwdcnt+A[current_counter]-(tot/cn)
3.3 keyword의 입력개수로 ec를 리턴한다.
```

[0047] 필요에 따라 확장테이블을 해쉬테이블로 구성하여 표 2의 1에서 키워드를 보다 빠르게 검색할 수도 있다.

[0048] 도 4 내지 도 6은 종래기술에 따른 카운팅 방법의 성능과 본 발명의 일실시예에 따른 카운팅 방법의 성능을 비교하기 위한 도면들이다. 제1 그래프(400)와 제2 그래프(500)는 이 분야의 최신 연구결과에 따라 카운팅 방법의 성능을 실제로 측정한 결과를 나타내고 있으며, 제3 그래프(600)는 본 발명의 일실시예에 따른 카운팅 방법의 성능을 실제로 측정한 결과를 각각 나타내고 있다.

[0049] 그래프들에서 가로축은 실제 키워드의 개수를, 세로축은 각각의 알고리즘에 의해 추정된 키워드의 개수를 나타내고 있으며, 메모리는 모두 1 메가바이트를 사용하였다.

[0050] 이와 같이, 본 발명의 실시예들에 따르면, 매우 적은 양의 메모리만으로 멀티셋의 개별 원소의 개수를 카운팅할 수 있다.

[0051] 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 이상

의 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.

[0052] 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.

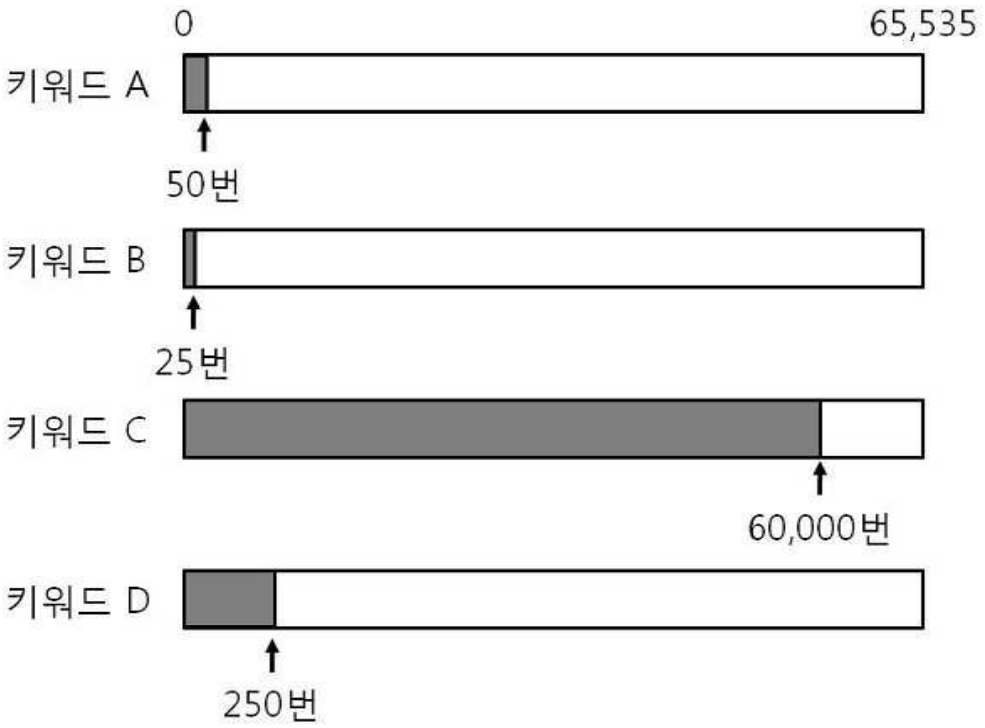
[0053] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic 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.

[0054] 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.

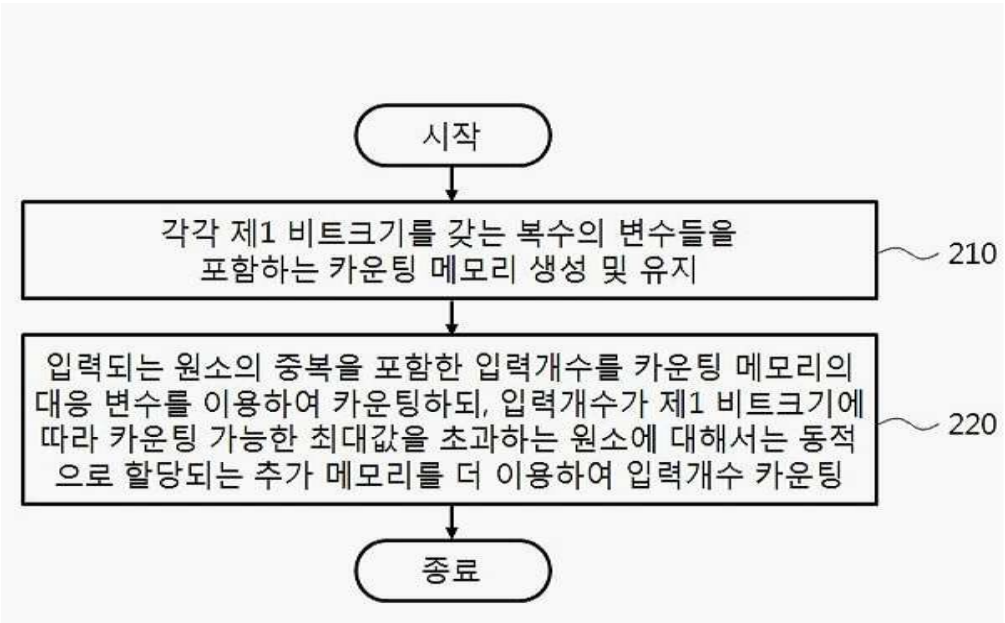
[0055] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

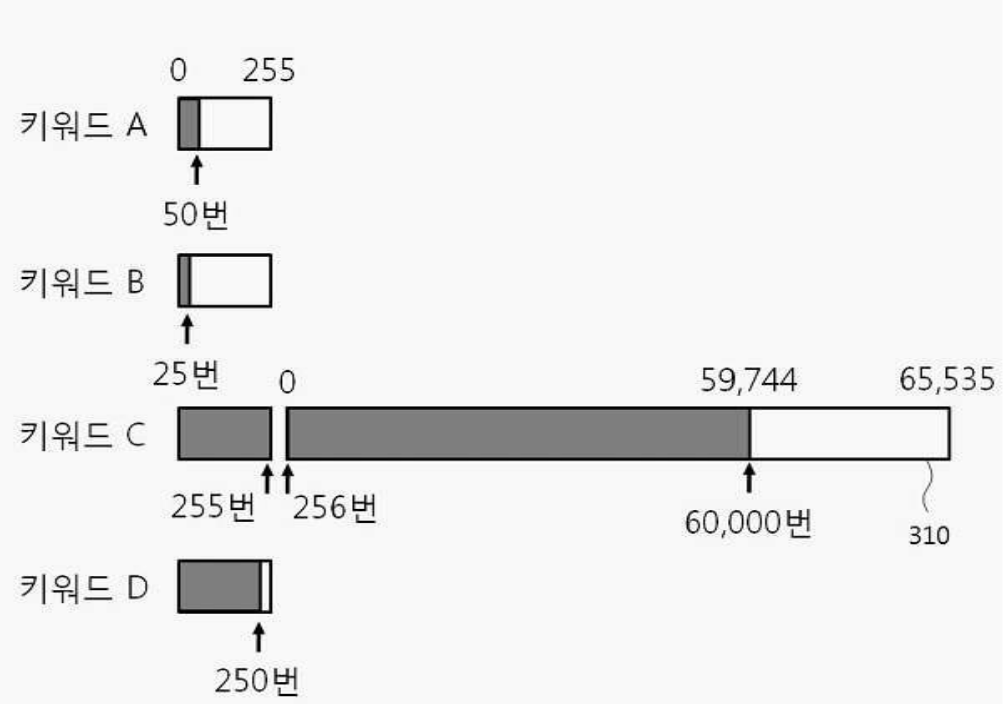
도면1


0
65,535
키워드 A
50번
키워드 B
25번
키워드 C
60,000번
키워드 D
250번

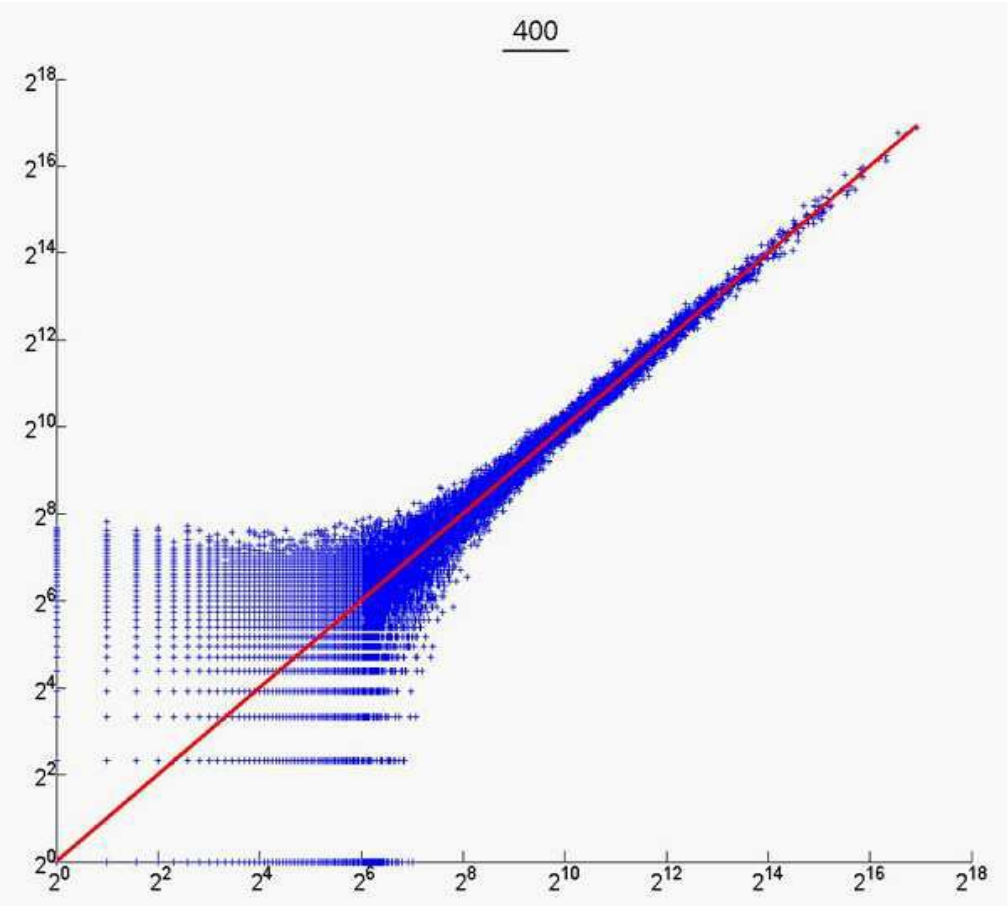
도면2


시작
각각 제1 비트크기를 갖는 복수의 변수들을 포함하는 카운팅 메모리 생성 및 유지
210
입력되는 원소의 중복을 포함한 입력개수를 카운팅 메모리의 대응 변수를 이용하여 카운팅하되, 입력개수가 제1 비트크기에 따라 카운팅 가능한 최대값을 초과하는 원소에 대해서는 동적으로 할당되는 추가 메모리를 더 이용하여 입력개수 카운팅
220
종료

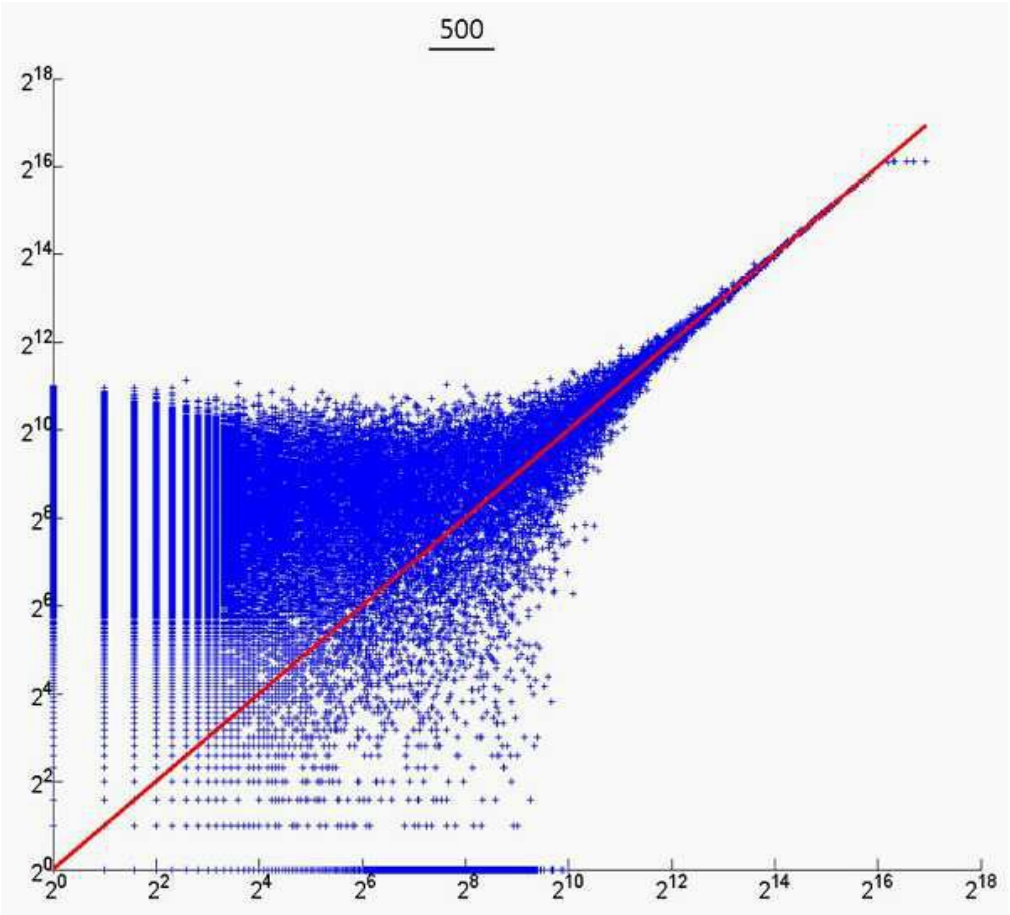
도면3


0
255
키워드 A
50번
키워드 B
25번
0
59,744
65,535
키워드 C
255번
256번
60,000번
310
키워드 D
250번

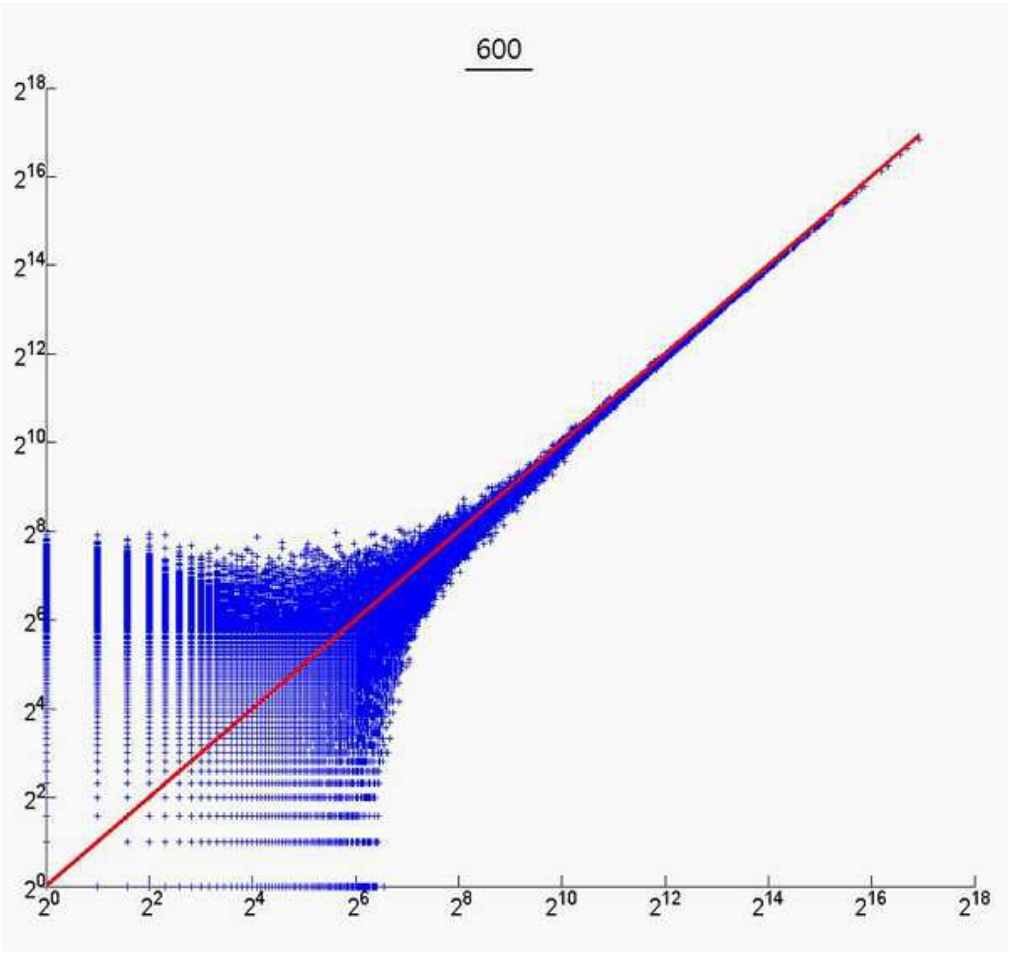
도면4


400
2^{18}
2^{16}
2^{14}
2^{12}
2^{10}
2^{8}
2^{6}
2^{4}
2^{2}
2^{0}
2^{0} 2^{2} 2^{4} 2^{6} 2^{8} 2^{10} 2^{12} 2^{14} 2^{16} 2^{18}

도면5


500
2^{18}
2^{16}
2^{14}
2^{12}
2^{10}
2^{8}
2^{6}
2^{4}
2^{2}
2^{0}
2^{0} 2^{2} 2^{4} 2^{6} 2^{8} 2^{10} 2^{12} 2^{14} 2^{16} 2^{18}

도면6


600
2^{18}
2^{16}
2^{14}
2^{12}
2^{10}
2^{8}
2^{6}
2^{4}
2^{2}
2^{0}
2^{0} 2^{2} 2^{4} 2^{6} 2^{8} 2^{10} 2^{12} 2^{14} 2^{16} 2^{18}