



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2019-0076657
(43) 공개일자 2019년07월02일

(51) 국제특허분류(Int. Cl.)

G06F 11/36 (2006.01)

(52) CPC특허분류

G06F 11/3604 (2013.01)

(21) 출원번호 10-2017-0178629

(22) 출원일자 2017년12월22일

심사청구일자 2017년12월22일

(71) 출원인

충남대학교산학협력단

대전광역시 유성구 대학로 99 (궁동, 충남대학교)

(72) 발명자

조은선

대전광역시 유성구 어은로 57, 110동 1504호

김영철

대전광역시 유성구 궁동로 18, 203호

(74) 대리인

홍성욱, 심경식

전체 청구항 수 : 총 14 항

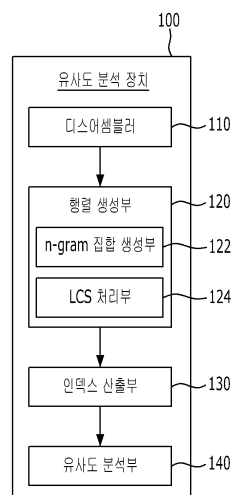
(54) 발명의 명칭 프로그램 유사도 분석 장치 및 그 방법

(57) 요약

프로그램 유사도 분석 장치 및 그 방법이 개시된다.

본 발명의 일 실시예에 따른 프로그램 유사도 분석 장치는, 제1 프로그램의 제1 함수정보와 제2 프로그램의 제2 함수정보에 n-gram을 적용하여 제1 n-gram 집합 및 제2 n-gram 집합을 생성하고, 상기 제1 n-gram 집합 및 제2 n-gram 집합을 이용하여 일정 크기의 행렬을 생성하는 행렬 생성부, 상기 행렬에서 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출하는 인덱스 산출부, 상기 산출된 인덱스에 기초하여 제1 함수 정보와 제2 함수 정보의 동일 여부를 판단하는 유사도 분석부를 포함한다.

대표도 - 도1



이 발명을 지원한 국가연구개발사업

과제고유번호 2015-0-00930

부처명 미래창조과학부

연구관리전문기관 정보통신기술진흥센터

연구사업명 SW전문인력양성사업/정보통신창의인재양성

연구과제명 SW중심대학(충남대)

기 여 율 1/1

주관기관 충남대학교 산학협력단

연구기간 2017.03.01 ~ 2017.12.31

명세서

청구범위

청구항 1

제1 프로그램의 제1 함수정보와 제2 프로그램의 제2 함수정보에 n-gram을 적용하여 제1 n-gram 집합 및 제2 n-gram 집합을 생성하고, 상기 제1 n-gram 집합 및 제2 n-gram 집합을 이용하여 일정 크기의 행렬을 생성하는 행렬 생성부;

상기 행렬에서 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출하는 인덱스 산출부; 및

상기 산출된 인덱스에 기초하여 제1 함수정보와 제2 함수정보의 동일 여부를 판단하는 유사도 분석부를 포함하는,

프로그램 유사도 분석 장치.

청구항 2

제1항에 있어서,

상기 제1 프로그램 또는 제2 프로그램으로부터 적어도 하나 이상의 함수와 그 함수의 명령어들로 구성된 제1 함수정보 또는 제2 함수정보를 함수단위로 각각 추출하는 디어셈블러를 더 포함하는 프로그램 유사도 분석 장치.

청구항 3

제1항에 있어서,

상기 행렬 생성부는,

상기 제1 함수정보와 제2 함수정보에서 각각 함수를 구성하는 명령어를 n-gram 단위로 분할하여 n개의 명령어로 구성된 제1 n-gram 집합 및 제2 n-gram 집합을 생성하는 n-gram 집합 생성부;

상기 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 제1 n-gram 집합과 제2 n-gram 집합의 쌍으로 이루어진 행렬을 생성하는 LCS 처리부를 포함하는 것을 특징으로 하는 프로그램 유사도 분석 장치.

청구항 4

제3항에 있어서,

상기 n-gram 집합 생성부는,

상기 제1 함수정보를 구성하는 명령어 갯수와 제2 함수정보를 구성하는 명령어 갯수가 다른 경우, 제1 함수정보와 제2 함수정보의 명령어 갯수를 동일하게 맞춘 후, n-gram 집합을 생성하는 것을 특징으로 하는 프로그램 유사도 분석 장치.

청구항 5

제3항에 있어서,

상기 LCS 처리부는,

상기 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 두 함수가 공통으로 갖는 명령어 중에서 가장 긴 열의 길이를 LCS 길이로 찾고, 각각의 원소들 간의 LCS 길이를 n으로 나눈 값을 원소로 갖는 정방행렬을 생성하는 것을 특징으로 하는 프로그램 유사도 분석 장치.

청구항 6

제3항에 있어서,

상기 인덱스 산출부는,

상기 행렬에 헝가리안 알고리즘(Hungarian algorithm)을 적용하여 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출하는 것을 특징으로 하는 프로그램 유사도 분석 장치.

청구항 7

제1항에 있어서,

상기 유사도 분석부는

기울기가 '1'인 인덱스가 연속되는 빈도(갯수)에 기초하여 상기 제1 함수정보와 제2 함수정보의 유사도를 산출하는 것을 특징으로 하는 프로그램 유사도 분석 장치.

청구항 8

제7항에 있어서,

상기 유사도 분석부는,

$y = x$ 의 관계에 대한 인덱스의 갯수 또는 인덱스 간 기울기 1을 유지하는 연속 인덱스의 빈도를 이용하여 유사도를 산출하는 것을 특징으로 하는 프로그램 유사도 분석 장치.

청구항 9

프로그램의 유사도를 분석하는 장치가 프로그램의 유사도를 분석하는 방법에 있어서,

제1 프로그램의 제1 함수정보와 제2 프로그램의 제2 함수정보에 n-gram을 적용하여 제1 n-gram 집합 및 제2 n-gram 집합을 생성하고, 상기 제1 n-gram 집합 및 제2 n-gram 집합을 이용하여 일정 크기의 행렬을 생성하는 단계;

상기 행렬에서 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출하는 단계; 및

상기 산출된 인덱스에 기초하여 제1 함수정보와 제2 함수정보의 동일 여부를 판단하는 단계를 포함하는,

프로그램 유사도 분석 방법.

청구항 10

제9항에 있어서,

상기 행렬을 생성하는 단계 이전에,

상기 제1 프로그램 또는 제2 프로그램으로부터 적어도 하나 이상의 함수와 그 함수의 명령어들로 구성된 제1 함수정보 또는 제2 함수정보를 함수단위로 각각 추출하는 단계를 더 포함하는 프로그램 유사도 분석 방법.

청구항 11

제9항에 있어서,

상기 행렬을 생성하는 단계는,

상기 제1 함수정보와 제2 함수정보에서 각각 함수를 구성하는 명령어를 n-gram 단위로 분할하여 n개의 명령어로 구성된 제1 n-gram 집합 및 제2 n-gram 집합을 생성하는 단계; 및

상기 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 제1 n-gram 집합과 제2 n-gram 집합의 쌍으로 이루어진 행렬을 생성하는 단계를 포함하는 것을 특징으로 하는 프로그램 유사도 분석 방법.

청구항 12

제11항에 있어서,

상기 제1 n-gram 집합 및 제2 n-gram 집합을 생성하는 단계에서,

상기 제1 함수정보를 구성하는 명령어 갯수와 제2 함수정보를 구성하는 명령어 갯수가 다른 경우, 제1 함수정보와 제2 함수정보의 명령어 갯수를 동일하게 맞춘 후, n-gram 집합을 생성하는 것을 특징으로 하는 프로그램 유사도 분석 방법.

청구항 13

제9항에 있어서,

상기 인덱스를 산출하는 단계는,

상기 행렬에 헝가리안 알고리즘(Hungarian algorithm)을 적용하여 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출하는 것을 특징으로 하는 프로그램 유사도 분석 방법.

청구항 14

제9항에 있어서,

상기 동일 여부를 판단하는 단계는,

기울기가 '1'인 인덱스가 연속되는 빈도(갯수)에 기초하여 상기 제1 함수정보와 제2 함수정보의 유사도를 산출하는 것을 특징으로 하는 프로그램 유사도 분석 방법.

발명의 설명

기술 분야

[0001] 본 발명은 프로그램 유사도 분석 장치 및 그 방법에 관한 것으로, 특히 오픈소스 프로그램으로부터 추출한 함수 정보와 분석 대상 프로그램으로부터 추출한 함수정보의 명령어 리스트 간에 유사도를 분석하여 동일 함수를 검출하는 프로그램 유사도 분석 장치 및 그 방법에 관한 것이다.

배경 기술

[0002] 바이너리 프로그램 분석은 소스 코드에 의존하지 않고 프로그램의 구조와 실행 흐름 등을 파악하는데 매우 중요하다. 하지만 다른 모든 분석들과 마찬가지로, 정확한 프로그램 분석에는 프로그램의 크기에 비례하여 많은 시간이 걸리게 되고, 시간이 적게 소요되는 간단한 분석 방법으로는 그 적용 범위가 매우 작거나 잘못된 분석 결과를 도출할 확률이 높아진다. 따라서 바이너리 프로그램을 효율적으로 분석하려는 연구가 필요하다.

[0003] 바이너리 유사도 분석은 취약점 분석, 악성코드 분석, 표절 탐지 등에서 사용되고 있고, 바이너리 유사도 분석에 관한 연구는 여러 가지 방법으로 진행되고 있다. CoP는 원본 프로그램과 비교 대상 프로그램의 시퀀스를 비교하기 위해 기본 블록 단위로 의미를 정형화하고 기본 블록 간의 비교를 수행하여 의미적으로 같은 블록의 수를 계산하여 유사도를 측정하는 시스템이다. 기본 블록 단위로 비교를 시작하여 원본 프로그램의 기본 블록과 같지 않다고 판단될 경우에는 기본 블록의 분리/병합을 통해 반복적으로 비교를 수행한다. 그러나, 이 방법으로 프로그램 전체를 대상으로 분석을 수행하기 때문에 많은 시간이 소요된다.

[0004] 프로그램의 의미와 실행 흐름을 고려하기 위해 Graph-edit distance를 이용하여 함수를 탐지하는 방법도 존재한다. Graph-edit distance란 비교하는 두 그래프 사이의 유사도를 말한다. Graph-edit distance의 값은 그래프의 노드와 연결선에 관련된 여섯 가지 edit operator를 이용하여 계산된 블록간의 수정비용 행렬로부터 블록쌍들의 수정비용의 합이 가장 작은 것을 계산한다. 계산과정에서 시간 복잡도를 $O(n^3)$ 으로 줄여주는 Hungarian algorithm을 적용하여 계산 시간을 줄이지만 행렬의 크기가 커질수록 기하급수적으로 늘어난다. 즉, 크기가 큰 함수에 대해서는 적합한 방법이 아니다.

[0005] IDA Pro의 FLIRT는 탐지할 라이브러리의 시그니처 파일에 함수의 첫 32바이트 패턴을 저장해두고 시그니처를 읽어 패턴을 비교하는 방법을 기초로 한다. 패턴 비교이기 때문에 탐지속도가 빠르다. 하지만 시그니처 파일을 적용할 바이너리에 코드 수정이나 사용된 라이브러리의 버전이 바뀌는 등 변화가 생기면 정확히 같지 않게 되기 때문에 탐지되는 함수의 수가 대폭 감소하게 된다. 따라서 정확한 비교를 위해서 탐지할 라이브러리의 시그니처 파일을 버전별로 생성해야하는 번거로운 일이 발생한다.

[0006] 상술한 바와 같이 바이너리 유사도 분석에 관한 연구는 여러 가지 방법으로 진행되고 있으나, 종래에는 동일 함수의 서로 다른 버전에 대한 유사도 분석에 대해서 별도로 이루어진 연구가 없었다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 한국 등록특허공보 제10-1482073호(2006.04.24.)

발명의 내용

해결하려는 과제

[0008] 본 발명의 목적은, 동일 함수의 서로 다른 버전에 대한 유사도를 분석할 수 있는 프로그램 유사도 분석 장치 및 그 방법을 제공하는 것이다.

[0009] 본 발명이 해결하고자 하는 과제는 이상에서 언급한 과제(들)로 제한되지 않으며, 언급되지 않은 또 다른 과제(들)는 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0010] 상기한 목적을 달성하기 위하여, 본 발명의 일 실시예에 따른 프로그램 유사도 분석 장치는, 제1 프로그램의 제1 함수정보와 제2 프로그램의 제2 함수정보에 n-gram을 적용하여 제1 n-gram 집합 및 제2 n-gram 집합을 생성하고, 상기 제1 n-gram 집합 및 제2 n-gram 집합을 이용하여 일정 크기의 행렬을 생성하는 행렬 생성부, 상기 행렬에서 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출하는 인덱스 산출부, 상기 산출된 인덱스에 기초하여 제1 함수정보와 제2 함수정보의 동일 여부를 판단하는 유사도 분석부를 포함한다.

[0011] 바람직하게는, 상기 프로그램 유사도 분석 장치는, 상기 제1 프로그램 또는 제2 프로그램으로부터 적어도 하나 이상의 함수와 그 함수의 명령어들로 구성된 제1 함수정보 또는 제2 함수정보를 함수단위로 각각 추출하는 디스어셈블러를 더 포함할 수 있다.

[0012] 바람직하게는, 상기 행렬 생성부는, 상기 제1 함수정보와 제2 함수정보에서 각각 함수를 구성하는 명령어를 n-gram 단위로 분할하여 n개의 명령어로 구성된 제1 n-gram 집합 및 제2 n-gram 집합을 생성하는 n-gram 집합 생성부, 상기 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 제1 n-gram 집합과 제2 n-gram 집합의 쌍으로 이루어진 행렬을 생성하는 LCS 처리부를 포함할 수 있다.

[0013] 바람직하게는, 상기 n-gram 집합 생성부는, 상기 제1 함수정보를 구성하는 명령어 갯수와 제2 함수정보를 구성하는 명령어 갯수가 다른 경우, 제1 함수정보와 제2 함수정보의 명령어 갯수를 동일하게 맞춘 후, n-gram 집합을 생성할 수 있다.

[0014] 바람직하게는, 상기 LCS 처리부는, 상기 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 두 함수가 공통으로 갖는 명령어 중에서 가장 긴 열의 길이를 LCS 길이로 찾고, 각각의 원소들 간의 LCS 길이를 n으로 나눈 값을 원소로 갖는 정방행렬을 생성할 수 있다.

[0015] 바람직하게는, 상기 인덱스 산출부는, 상기 행렬에 헝가리안 알고리즘(Hungarian algorithm)을 적용하여 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출할 수 있다.

[0016] 바람직하게는, 상기 유사도 분석부는 기울기가 '1'인 인덱스가 연속되는 빈도(갯수)에 기초하여 상기 제1 함수정보와 제2 함수정보의 유사도를 산출할 수 있다.

[0017] 바람직하게는, 상기 유사도 분석부는, $y = x$ 의 관계에 대한 인덱스의 갯수 또는 인덱스 간 기울기 1을 유지하는 연속 인덱스의 빈도를 이용하여 유사도를 산출할 수 있다.

[0018] 상기한 목적을 달성하기 위하여, 본 발명의 일 실시예에 따른 프로그램 유사도 분석 방법은, 프로그램의 유사도를 분석하는 장치가 프로그램의 유사도를 분석하는 방법에 있어서, 제1 프로그램의 제1 함수정보와 제2 프로그램의 제2 함수정보에 n-gram을 적용하여 제1 n-gram 집합 및 제2 n-gram 집합을 생성하고, 상기 제1 n-gram 집합 및 제2 n-gram 집합을 이용하여 일정 크기의 행렬을 생성하는 단계, 상기 행렬에서 행과 열을 중복으로 선택

하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출하는 단계, 상기 산출된 인덱스에 기초하여 제1 함수정보와 제2 함수정보의 동일 여부를 판단하는 단계를 포함한다.

[0019] 바람직하게는, 프로그램 유사도 분석 방법은, 상기 행렬을 생성하는 단계 이전에, 상기 제1 프로그램 또는 제2 프로그램으로부터 적어도 하나 이상의 함수와 그 함수의 명령어들로 구성된 제1 함수정보 또는 제2 함수정보를 함수단위로 각각 추출하는 단계를 더 포함할 수 있다.

[0020] 바람직하게는, 상기 행렬을 생성하는 단계는, 상기 제1 함수정보와 제2 함수정보에서 각각 함수를 구성하는 명령어를 n-gram 단위로 분할하여 n개의 명령어로 구성된 제1 n-gram 집합 및 제2 n-gram 집합을 생성하는 단계, 상기 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 제1 n-gram 집합과 제2 n-gram 집합의 쌍으로 이루어진 행렬을 생성하는 단계를 포함할 수 있다.

[0021] 바람직하게는, 상기 제1 n-gram 집합 및 제2 n-gram 집합을 생성하는 단계에서, 상기 제1 함수정보를 구성하는 명령어 갯수와 제2 함수정보를 구성하는 명령어 갯수가 다른 경우, 제1 함수정보와 제2 함수정보의 명령어 갯수를 동일하게 맞춘 후, n-gram 집합을 생성할 수 있다.

[0022] 바람직하게는, 상기 인덱스를 산출하는 단계는, 상기 행렬에 헝가리안 알고리즘(Hungarian algorithm)을 적용하여 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출할 수 있다.

[0023] 바람직하게는, 상기 동일 여부를 판단하는 단계는, 기울기가 '1인 인덱스가 연속되는 빈도(갯수)에 기초하여 상기 제1 함수정보와 제2 함수정보의 유사도를 산출할 수 있다.

발명의 효과

[0024] 본 발명에 따르면, 오프소스 프로그램의 함수정보와 분석 대상의 함수정보간의 유사도 분석을 수행하여 동일 함수를 분석대상에서 제외하여 효과적인 바이너리 분석을 수행할 수 있다.

[0025] 본 발명의 효과들은 이상에서 언급한 효과들로 제한되지 않으며, 언급되지 않은 또 다른 효과들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[0026] 도 1은 본 발명의 일 실시예에 따른 프로그램 유사도 분석 장치를 설명하기 위한 블록도이다.

도 2는 본 발명의 일 실시예에 따른 제1 n-gram 집합과 제2 n-gram 집합을 이용하여 행렬을 생성하는 방법을 설명하기 위한 예시도이다.

도 3은 본 발명의 일 실시예에 따른 행렬에 헝가리안 알고리즘을 적용한 결과를 설명하기 위한 예시도이다.

도 4는 본 발명의 일 실시예에 따른 두 함수의 유사도 분석 방법을 설명하기 위한 예시도이다.

도 5는 본 발명의 일 실시예에 따른 프로그램간 유사도 분석 방법을 설명하기 위한 순서도이다.

발명을 실시하기 위한 구체적인 내용

[0027] 본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세하게 설명하고자 한다. 그러나 이는 본 발명을 특정한 실시 형태에 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다. 각 도면을 설명하면서 유사한 참조부호를 유사한 구성요소에 대해 사용하였다.

[0028] 제1, 제2, A, B 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다. 예를 들어, 본 발명의 권리 범위를 벗어나지 않으면서 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소도 제1 구성요소로 명명될 수 있다. 및/또는 이라는 용어는 복수의 관련된 기재된 항목들의 조합 또는 복수의 관련된 기재된 항목들 중의 어느 항목을 포함한다.

[0029] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다.

- [0030] 본 출원에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [0031] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.
- [0032] 이하에서는 본 발명에 따른 바람직한 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.
- [0033] 도 1은 본 발명의 일 실시예에 따른 프로그램 유사도 분석 장치를 설명하기 위한 블록도, 도 2는 본 발명의 일 실시예에 따른 제1 n-gram 집합과 제2 n-gram 집합을 이용하여 행렬을 생성하는 방법을 설명하기 위한 예시도, 도 3은 본 발명의 일 실시예에 따른 행렬에 헵가리언 알고리즘을 적용한 결과를 설명하기 위한 예시도, 도 4는 본 발명의 일 실시예에 따른 두 함수의 유사도 분석 방법을 설명하기 위한 예시도이다.
- [0034] 도 1을 참조하면, 본 발명의 일 실시예에 따른 프로그램간 유사도 분석 장치(100) 디스어셈블러(disassembler)(110), 행렬 생성부(120), 인덱스 산출부(130), 유사도 분석부(140)를 포함한다.
- [0035] 디스어셈블러(110)는 제1 프로그램과 제2 프로그램으로부터 제1 함수정보 및 제2 함수정보를 추출한다. 여기서, 제1 프로그램은 기준 프로그램으로 오픈소스 바이너리 프로그램일 수 있고, 제2 프로그램은 분석대상이 되는 바이너리 프로그램일 수 있다. 또한, 제1 프로그램과 제2 프로그램은 서로 다른 버전의 프로그램일 수 있다.
- [0036] 프로그램은 함수들로 구성되고, 각 함수들은 명령어들로 구성된다. 따라서, 디스어셈블러(110)는 함수 단위로 명령어들을 추출한다. 따라서, 제1 함수정보는 적어도 하나 이상의 함수, 각 함수를 구성하는 명령어들을 포함하고, 제2 함수정보는 적어도 하나 이상의 함수, 각 함수를 구성하는 명령어들을 포함할 수 있다.
- [0037] 예를 들어, 제1 프로그램이 A함수, B함수, C함수로 구성된 경우, 디스어셈블러(110)는 A함수와 A함수를 구성하는 명령어(예컨대, A1, A2, A3)를 추출하고, B함수와 B함수를 구성하는 명령어(예컨대, B1, B2, B3, B4)를 추출하며, C함수와 C함수를 구성하는 명령어(예컨대, C1, C2, C3)를 추출한다. 이때, 추출된 A함수와 A함수와 A함수를 구성하는 명령어, B함수와 B함수를 구성하는 명령어, C함수와 C함수를 구성하는 명령어가 함수정보일 수 있다.
- [0038] 디스어셈블러(110)로는 대표적으로 IDApro 디스어셈블러가 이용될 수 있으나, 함수가 동작하는 프로세서와 하드웨어 플랫폼에 따라서 목적에 부합되는 기타 다양한 디스어셈블러가 이용될 수 있다.
- [0039] 한편, 제1 프로그램으로부터 추출된 제1 함수정보가 미리 데이터베이스에 저장된 경우, 제1 프로그램으로부터 제1 함수정보를 추출하는 동작은 생략 가능하다. 또한, 제1 프로그램과 제2 프로그램으로부터 추출된 제1 함수정보 및 제2 함수정보가 미리 데이터베이스에 저장된 경우, 디스어셈블러(110)는 생략 가능하다.
- [0040] 행렬 생성부(120)는 제1 프로그램의 제1 함수정보와 제2 프로그램의 제2 함수정보에 n-gram을 적용하여 제1 n-gram 집합 및 제2 n-gram 집합을 생성하고, 제1 n-gram 집합 및 제2 n-gram 집합을 이용하여 일정 크기의 행렬을 생성한다. 이때, 제1 함수정보와 제2 함수정보는 하나 이상의 함수와 각 함수의 명령어들을 포함하므로, 행렬 생성부(120)는 함수 단위로 하나 이상의 행렬을 생성할 수 있다. 이하에서는 설명의 편의를 위해 제1 함수정보와 제2 함수정보가 하나의 함수로 이루어진 것으로 가정하여 설명하기로 한다.
- [0041] 이러한 행렬 생성부(120)는 n-gram 집합 생성부(122), LCS 처리부(124)를 포함한다.
- [0042] n-gram 집합 생성부(122)는 제1 함수정보와 제2 함수정보를 구성하는 명령어를 n-gram 단위로 분할하여 제1 n-gram 집합 및 제2 n-gram 집합을 생성한다. 여기서, n-gram이란 주어진 문자열로부터 얻을 수 있는 연속된 n개의 문자열을 말한다. 따라서, n-gram 집합 생성부(122)는 제1 함수정보에 포함된 명령어들을 n개의 명령어들로 분할하여 n개의 명령어로 구성된 제1 n-gram 집합을 생성하고, 제2 함수정보에 포함된 명령어들을 n개의 명령어들로 분할하여 n개의 명령어로 구성된 제2 n-gram 집합을 생성한다. n-gram을 사용하면, 명령어 리스트를 그대로 사용하는 것보다 더 많은 유연성을 허용한다. 이때, 니모닉 전체 시퀀스의 쌍은 높은 유사성을

위해 동일할 필요는 없다.

- [0043] 예를 들어, 제1 함수정보가 {push, mov, push, push, ret}, 제2 함수정보가 {push, mov, push, pop, ret}인 경우, 3-gram 집합을 생성하는 경우에 대해 설명하기로 한다. 제1 함수정보가 {push, mov, push, push, ret}이므로, n-gram 집합 생성부(122)는 제1 함수정보를 3개의 명령어들로 구성된 {push, mov, push}, {mov, push, push}, {push, push, ret} 총 3개의 명령어 집합으로 구성된 제1 n-gram 집합을 생성할 수 있다. 또한, 제2 함수정보가 {push, mov, push, pop, ret}이므로, n-gram 집합 생성부(122)는 제2 함수정보를 3개의 명령어들로 구성된 {push, mov, push}, {mov, push, pop}, {push, pop, ret} 총 3개의 명령어 집합으로 구성된 제2 n-gram 집합을 생성할 수 있다.
- [0044] 또한, n-gram 집합 생성부(122)는 제1 함수정보를 구성하는 명령어 갯수와 제2 함수정보를 구성하는 명령어 갯수가 다른 경우, 제1 함수정보와 제2 함수정보의 명령어 갯수를 동일하게 맞춘 후, n-gram 집합을 생성할 수 있다. 이때, n-gram 집합 생성부(122)는 명령어 갯수가 더 적은 개수로 맞출 수 있다. 따라서, n-gram 집합 생성부(122)는 갯수가 더 많은 명령어 리스트에서 더 많은 개수만큼 명령어를 잘라낸 후 n-gram 집합을 생성할 수 있다. 예컨대, n-gram 집합 생성부(122)는 개수가 많은 명령어에서 마지막 명령어, 최초 명령어 등 다양한 위치의 명령어를 잘라낼 수 있으나, 바람직하게는 마지막 명령어 순으로 제거할 수 있다.
- [0045] 예를 들어, 제1 함수정보가 {push, mov, push, push, pop, ret}이고, 제2 함수정보가 {push, mov, push, pop, ret}이며, 3-gram 집합을 생성하는 경우에 대해 설명하기로 한다. 이 경우 제1 함수정보를 구성하는 명령어는 6개이고, 제2 함수정보를 구성하는 명령어는 5개이므로, 제1 함수정보를 구성하는 명령어를 제2 함수정보를 구성하는 명령어의 개수에 해당하는 5개로 맞춰야 한다. 이를 위해 n-gram 집합 생성부(122)는 제1 함수정보에서 마지막 명령어에 해당하는 'ret'를 잘라낸다. 그러면, 제1 함수정보는 {push, mov, push, push, pop}로 변경되고, n-gram 집합 생성부(122)는 {push, mov, push}, {mov, push, push}, {push, push, pop}으로 구성된 제1 n-gram 집합을 생성할 수 있다.
- [0046] LCS 처리부(124)는 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 정방행렬($m \times m$ 행렬)을 생성한다. LCS 알고리즘은 두 열(Sequence) S1과 S2가 주어졌을 때, 두 열에 공통으로 들어있는 부분열 중에서 가장 긴 열의 길이를 찾아내는 것이다. 여기서 부분열이란 다른 문자열에서 몇몇 문자가 빠져있어도 순서는 바뀌지 않은 문자열을 말한다.
- [0047] 따라서, LCS 처리부(124)는 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 두 함수가 공통으로 갖는 명령어 중에서 가장 긴 열의 길이를 LCS 길이로 찾고, 각각의 원소들 간의 LCS의 길이를 n으로 나눈 값을 원소로 갖는 $m \times m$ 행렬을 생성한다. 이때 임계값이 있는 경우, 임계값을 넘지 않는 원소들은 0으로 바꾸어 행렬을 완성할 수 있다.
- [0048] 또한, LCS 처리부(124)는 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 두 함수가 공통으로 갖는 명령어 리스트를 추출하는데, 추출된 명령어 리스트의 길이를 두 함수 중에서 더 긴 명령어 리스트의 크기로 나눈 값을 LCS 유사도 값으로 선정할 수 있다. 짧은 명령어 리스트의 길이로 나누게 되면 함수의 크기가 많이 차이가 나게 되는 경우에도 유사도가 높게 측정될 가능성이 높기 때문이다.
- [0049] 예를 들어, 도 2의 (a)와 같은 제1 n-gram 집합과 제2 n-gram 집합이 주어진 경우, 행렬을 생성하는 방법에 대해 설명하기로 한다. 이 경우, (a)에 LCS 알고리즘을 적용하면, 제1 n-gram 집합과 제2 n-gram 집합의 쌍으로 이루어진 (b)와 같은 행렬이 생성된다. 그러면, LCS 처리부(124)는 각 행렬 원소들간의 LCS 길이를 n-gram의 n에 해당하는 수(예컨대, 3-gram의 경우 3임)로 나누어 최종 행렬을 생성하게 된다.
- [0050] 인덱스 산출부(130)는 행렬 생성부(120)에서 생성된 행렬에서 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출한다. 이때, 인덱스 산출부(130)는 행렬에 헝가리안 알고리즘(Hungarian algorithm)을 적용하여 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출할 수 있다. 헝가리안 알고리즘은 $m \times m$ 매칭 문제를 해결하는 알고리즘으로, 정합문제(Assignment Problem)에서 Brute Force 방식으로 해결할 때 $O(n!)$ 시간복잡도를 $O(n^3)$ 으로 줄일 수 있다.
- [0051] 또한, 인덱스 산출부(130)는 행렬에 Hungarian algorithm을 적용하여 n-gram 쌍들의 가장 큰 원소의 합을 m으로 나눈 값을 n-gram 유사도로 산출할 수 있다. 여기서, m은 행렬의 크기일 수 있다.
- [0052] 예를 들어, 도 3의 (a)와 같은 행렬이 주어진 경우, 헝가리안 알고리즘을 적용하면, (b)와 같이 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출할 수 있다. 이때, n-gram 유사도는 $(1 + 2/3 +$

$2/3)/3 = 7/9$ 로 산출될 수 있다.

- [0053] 유사도 분석부(140)는 인덱스 산출부(130)에서 산출된 인덱스에 기초하여 제1 함수정보와 제2 함수정보의 동일 여부를 판단한다. 이때, 유사도 분석부(140)는 기울기 '1'인 인덱스가 연속되는 빈도(갯수)를 이용하여 제1 함수정보와 제2 함수정보의 동일 여부를 판단할 수 있다.
- [0054] 구체적으로, 유사도 분석부(140)는 $y = x$ 의 관계에 대한 인덱스의 개수 또는 인덱스 간 기울기 1을 유지하는 연속 인덱스의 빈도 등을 유사도를 산출할 수 있다. 또한, 유사도 분석부(140)는 $y = x$ 의 관계에 대한 인덱스의 개수와 인덱스 간 기울기 1을 유지하는 연속 인덱스의 빈도 중 더 큰 값을 유사도로 산출할 수 있다.
- [0055] 먼저, 도 4의 (a)를 참조하여 $y = x$ 의 관계에 대한 인덱스의 개수를 이용하여 유사도를 분석하는 방법에 대해 설명하기로 한다. 이 방법은 $n \times n$ 행렬일 때, 헝가리언 알고리즘에 의해 선택된 행렬의 인덱스 중 $y=x$ 그래프 상에 위치하는 인덱스의 비율을 유사도 값으로 설정하는 방법이다. $y=x$ 그래프 상에 위치하는 인덱스의 수를 m 이라고 표현할 때, 유사도는 m/n 이 될 수 있다. 여기서, m 의 최대값은 n 일 수 있다.
- [0056] (a)에서 볼 수 있듯이 헝가리 알고리즘에 의해 계산된 행렬 인덱스의 좌표 (0,0)에서 기울기가 1인 인덱스의 수가 많을수록 비교할 두 함수가 동일 할 확률이 높다.
- [0057] $n \times n$ 행렬의 경우, 유사도 분석부(140)는 수학식 1을 이용하여 유사도를 산출할 수 있다.

수학식 1

$$\text{similarity} = \frac{\# \text{ of a slope between index and (0,0) is 1}}{n}$$

- [0058]
- [0059] (a)에서 $y = x$ 의 관계에 대한 인덱스가 5개 이므로, 수학식 1에 따라 (a)의 유사도를 계산하면, $5/5 = 1$ 로 산출될 수 있고, 유사도가 '1'이므로, 제1 함수정보와 제2 함수정보는 동일하다는 것을 알 수 있다.
- [0060] 다음으로, 도 4의 (b)를 참조하여, 인덱스 간 기울기 1을 유지하는 연속 인덱스의 빈도를 이용하여 유사도를 분석하는 방법에 대해 설명하기로 한다. 이 방법은 $n \times n$ 행렬일 때, 헝가리언 알고리즘에 의해 선택된 행렬의 인덱스 중에서 연속된 m 개의 인덱스가 서로 기울기 1을 나타내는 경우(이러한 경우의 수를 k 로 표현)의 비율을 유사도 값으로 설정하는 방법이다. 이 경우 유사도는 $k/(n-m+1)$ 이 될 수 있다.
- [0061] (b)에서와 같이 헝가리 알고리즘에 의해 계산된 행렬의 인덱스가 주어지면, 인덱스 간 기울기가 '1'이 되는 빈도가 높을수록 비교할 두 함수가 동일할 확률이 높아진다는 것을 알 수 있다. $n \times n$ 행렬과 m 색인이 연속적으로 1의 기울기를 유지하는 경우, 유사도 분석부(140)는 아래 기재된 수학식 2를 이용하여 유사도를 산출할 수 있다.

수학식 2

$$\text{similarity} = \frac{\# \text{ of slope of } m \text{ consecutive indices is 1}}{n - m + 1}$$

- [0062]
- [0063] 여기서, $m = 2$ 일 때 유사도는 $5 / (7-2 + 1) = 5/6$ 일 수 있다. 또한, $m = 3$ 의 경우, 유사도는 $4 / (7-3 + 1) = 4/5$ 일 수 있다.
- [0064] 상술한 바와 같이 유사도 분석부(140)는 인덱스 산출부(130)에서 산출된 인덱스에 기초하여 두 프로그램에서 동일한 함수를 검출할 수 있다. 검출된 동일 함수를 분석 대상에서 제외함으로써, 바이너리 분석을 효과적으로 수행할 수 있고, 프로그램 분석의 양을 줄일 수 있다.
- [0065] 상술한 바와 같은 구성을 갖는 유사도 분석 장치(100)는 비교할 2개의 함수로부터 두개의 n -gram을 추출하고, n -gram간에 LCS를 적용하여 행렬을 생성하며, 헝가리 알고리즘을 통해 행렬에서 가장 유사한 n -gram 간의 매칭 관계를 찾을 수 있다. 이를 통해 유사도 분석 장치는 서로 다른 버전의 함수에 대한 유사성을 분석할 수 있다. 다시 말하면, 유사도 분석 장치(100)는 n -gram, LCS, Hungarian 알고리즘을 사용하여 두 함수 간의 유사성을 판단

할 수 있다.

- [0066] 이러한 유사도 분석 장치(100)는 컴파일러, 컴파일 옵션 및 버전으로 인한 기본 블록 또는 명령어 순서의 변경에 대한 재발 대응을 지원할 수 있고, 함수를 구성하는 명령어와 명령어 실행 흐름을 포함하는 함수의 의미를 고려할 수 있다.
- [0067] 한편, 유사도 분석 장치(100)는 개인용 컴퓨터(예를 들어, 데스크탑 컴퓨터, 노트북 컴퓨터, 태블릿 컴퓨터, 팜톱 컴퓨터 등), 워크스테이션, PDA, 웹 패드, 이동 전화기 등과 같은 디지털 기기에 포함되거나 연결될 수 있을 것이다.
- [0068] 도 5는 본 발명의 일 실시예에 따른 프로그램간 유사도 분석 방법을 설명하기 위한 순서도이다.
- [0069] 도 5를 참조하면, 유사도 분석 장치는 제1 프로그램과 제2 프로그램으로부터 제1 함수정보 및 제2 함수정보를 추출한다(S510). 이때 유사도 분석 장치는 바이너리 프로그램에서 함수 단위로 명령어들을 추출할 수 있다.
- [0070] 단계 S510이 수행되면, 유사도 분석 장치는 제1 함수정보와 제2 함수정보에 n-gram을 적용하여 제1 n-gram 집합 및 제2 n-gram 집합을 생성하고(S520), 제1 n-gram 집합 및 제2 n-gram 집합을 이용하여 일정 크기의 행렬을 생성한다(S530). 즉, 유사도 분석 장치는 제1 함수정보와 제2 함수정보에서 각각 함수를 구성하는 명령어를 n-gram 단위로 분할하여 제1 n-gram 집합 및 제2 n-gram 집합을 생성한다. 그런 후, 유사도 분석 장치는 제1 n-gram 집합 및 제2 n-gram 집합에 LCS 알고리즘을 적용하여 두 함수가 공통으로 갖는 명령어 중에서 가장 긴 열의 길이를 LCS 길이로 찾고, 각각의 원소들 간의 LCS의 길이를 n으로 나눈 값을 원소로 갖는 $m \times m$ 행렬을 생성할 수 있다.
- [0071] 단계 S530이 수행되면, 유사도 분석 장치는 생성된 행렬에서 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출한다(S540). 이때, 유사도 분석 장치는 행렬에 헝가리안 알고리즘(Hungarian algorithm)을 적용하여 행과 열을 중복으로 선택하지 않는 요소의 최대 합계를 구성하는 인덱스를 산출할 수 있다.
- [0072] 단계 S540이 수행되면, 유사도 분석 장치는 산출된 인덱스에 기초하여 제1 함수정보와 제2 함수정보의 동일 여부를 판단한다(S550). 이때, 유사도 분석 장치는 기울기 '1'인 인덱스가 연속되는 빈도(갯수)를 이용하여 두 함수의 동일 여부를 판단할 수 있다. 구체적으로, 유사도 분석 장치는 $y = x$ 의 관계에 대한 인덱스의 개수를 사용하는 방법 또는 인덱스 간 기울기 1을 유지하는 연속 인덱스의 빈도를 사용하는 방법을 이용하여 유사도를 산출할 수 있다.
- [0073] 유사도 분석 장치는 기울기가 '1'인 인덱스의 수가 많을수록 두 함수가 동일할 확률이 높다고 판단할 수 있다.
- [0074] 이하, 본 발명에서 적용한 방법의 성능에 대해 설명하기로 한다.
- [0075] 본 발명에서 적용한 방법의 성능을 측정하기 위해 오픈 소스 라이브러리인 OpenSSL 1.0.1f 버전과 1.0.2h 버전의 두 함수에 대한 실험을 수행하였다. 비교하는 두 함수가 1.0.1f 버전의 경우 이름이 존재하는 함수는 1757개, 1.0.2h 버전의 경우 1783개로 총 3,132,731개의 함수 쌍에 대해서 분석을 수행하였다. 두 dll 파일로부터 추출되는 이름이 존재하는 함수 중에서 같은 이름을 갖는 함수들은 서로 같다고 추정하였다. 두 dll 사이의 같은 이름을 갖는 함수 쌍은 1699개 즉, 같은 함수는 1699개로 생각하고 실험하였다. 전체 분석 결과에 코사인 유사도 값과 명령어 LCS 값을 0.7부터 1까지 0.05 단위로 필터링하여 잘못된 분석의 비율과 잘못된 분석 중에서도 치명적인 결과를 가져올 수 있는 False Positive의 비율을 적절하게 낮추는 필터 값을 선택할 수 있도록 한다.
- [0076] 표 1은 코사인 유사도와 명령어 LCS 값에 따른 필터링을 적용하였을 때의 False Positive와 False Negative 비율과 절대수치를 나타낸다.

표 1

Cosine	LCS	FN	FN 비율	FP	FP 비율	F	F 비율
0.7	0.7	207	0.006608	8744	0.279117	8951	0.285725
0.7	0.75	281	0.00897	5398	0.17231	5679	0.18128
0.7	0.8	411	0.01312	3685	0.117629	4096	0.130749
0.7	0.85	622	0.019855	2710	0.086506	3332	0.106361
0.7	0.9	895	0.028569	2074	0.066204	2969	0.094774
1	0.8	835	0.026654	2535	0.08092	3370	0.107574
1	0.85	914	0.029176	2192	0.069971	3106	0.099147
1	0.9	1049	0.033485	1806	0.057649	2855	0.091135
1	0.95	1315	0.041976	607	0.019376	1922	0.061352
1	1	1449	0.046254	496	0.015833	1945	0.062086

[0077]

[0078] 바이너리 분석을 할 때 False Positive의 비율이 높을수록 치명적인 결과가 도출될 수 있다. False Positive의 비율은 코사인 유사도 값과 명령어 LCS 값이 모두 1일 때 가장 낮았다. 그러나 False Positive의 비율을 최소로 낮추는 코사인 유사도와 명령어 LCS의 값은 오히려 False Positive와 False Negative 합인 False의 비율을 증가시킬 수 있다. 코사인 유사도 값이 1이고 명령어 LCS의 값이 0.95일 때, False 비율이 가장 낮았다.

[0079] 표 2는 명령어 LCS와 n-gram 기법에 대해 비교한 결과이다.

표 2

함수 이름	명령어	LCS값	시간(sec)	n-gram	시간(sec)
CAST_set_key	약 1000	0.73674	0.40603	0.82839	6637.43
BN_mod_sqrt	약 600	0.7874	0.21035	0.88074	778.825
ASN1_item_ex_d2i	약 600	0.77997	0.12318	0.80202	632.651
RAND_poll	약 600	0.81938	0.14942	0.73683	497.605
DES_ede3_cfb_encrypt	약 600	0.7376	0.1973	0.66712	911.342

[0080]

[0081] 명령어 LCS, n-gram 기법 모두 같다고 추정되는 함수 쌍들에 대한 유사도 값은 크게 차이나지 않았지만 분석 시간에서 크게 차이가 나는 것을 볼 수 있다. 시간 효율적인 측면에서 보았을 때, 명령어 LCS 기법이 n-gram 기법에 비해 좋다고 볼 수 있다.

[0082] 결과적으로 코사인 유사도와 명령어 LCS 값에 따른 precision과 recall을 기반으로 F-measure를 계산하여 적절한 코사인 유사도와 명령어 LCS의 값을 선정한다. precision은 Positive로 판정된 것 중에 True Positive가 차지하는 비율을 나타내고 recall은 실제로 같은 함수 중에서 Positive로 판정된 비율을 말한다. F-measure는 가중치를 적용한 precision과 recall에 대한 조화 평균값이다.

[0083] 표 3은 코사인 유사도와 명령어 LCS 값에 따른 F-measure를 나타낸다.

표 3

Cosine	LCS	precision	recall	F-measure
0.7	0.95	0.39	0.265	0.346709
0.75	0.95	0.39	0.265	0.346709
0.8	0.95	0.39	0.265	0.346709
0.85	0.95	0.39	0.265	0.346709
0.9	0.95	0.389	0.264	0.345675
0.95	0.95	0.388	0.263	0.344641
0.85	0.85	0.294	0.619	0.341456
0.8	0.85	0.292	0.624	0.339866
0.85	0.8	0.284	0.732	0.338904
0.9	0.85	0.291	0.605	0.337346

[0084]

[0085] F-measure는 precision에 가중치를 두고 계산하였다. 명령어 LCS의 값이 0.95이고 코사인 유사도 값이 0.7일 때, F-measure 값이 가장 높은 유사도를 보이는 것으로 보아 필터링은 코사인 유사도와 명령어 LCS 값을 각각 0.95, 0.7로 선정하는 것이 가장 좋다는 판단을 하였다.

[0086] 본 발명에서는 바이너리 유사도를 분석하는 코사인 유사도, 명령어 LCS, n-gram matrix & Hungarian algorithm을 적용하여 그 효율성에 대해 연구하였다.

[0087] 본 발명에서는 OpenSSL DLL 파일의 두 가지 버전 (1.0.1f 및 1.0.2h)의 함수에 대한 실험을 수행하였다. 그러나, 이 파일은 안전하다고 간주되어 종종 의심스러운 코드로 나타나기도 한다. 2개의 n-gram 집합의 LCS에 헝가리 알고리즘만을 적용할 때 $y = x$ 의 관계에 대한 인덱스의 개수를 사용하는 방법 또는 인덱스 간 기울기 1을 유지하는 연속 인덱스의 빈도를 사용하는 방법을 적용하지 않고 F-measure(즉, 정밀도와 회수율의 고조파 평균)는 0.329 이하이다. 그러나 본 발명에 따른 전체 단계를 완료할 때 F-measure 값은 최대 0.517에 이를 수 있다.

[0088] 본 발명에 따른 방법은 필요한 정보의 양과 분석 시간을 줄이고 니모닉(mnemonics) 순서를 블록 단위로 수정하더라도 견고함을 기대할 수 있다. 따라서, 본 발명은 맬웨어 탐지, 취약한 코드 탐지, 코드 도난 탐지 등에 대한 바이너리 분석 시간을 효과적으로 절약할 수 있다.

[0089] 이제까지 본 발명에 대하여 그 바람직한 실시예들을 중심으로 살펴보았다. 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

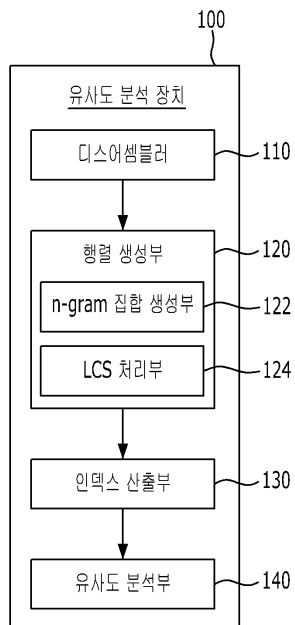
부호의 설명

[0090]

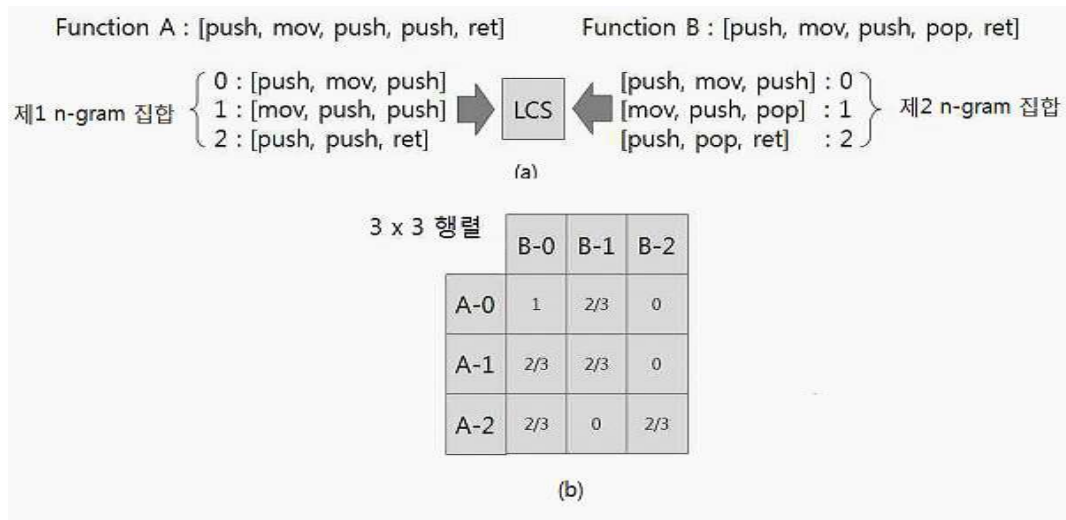
- 100 : 프로그램 유사도 분석 장치
- 110 : 디스어셈블러
- 120 : 행렬 생성부
- 130 : 인덱스 산출부
- 140 : 유사도 분석부

도면

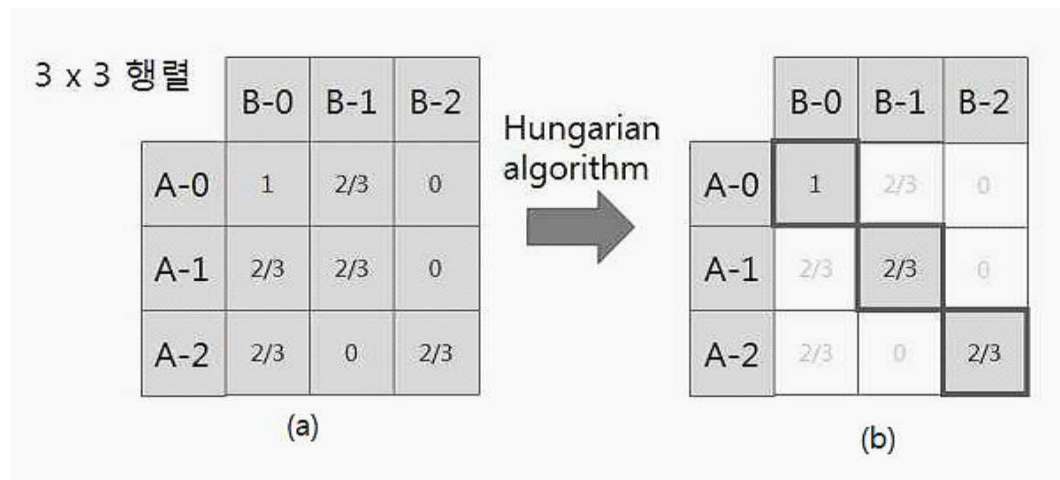
도면1



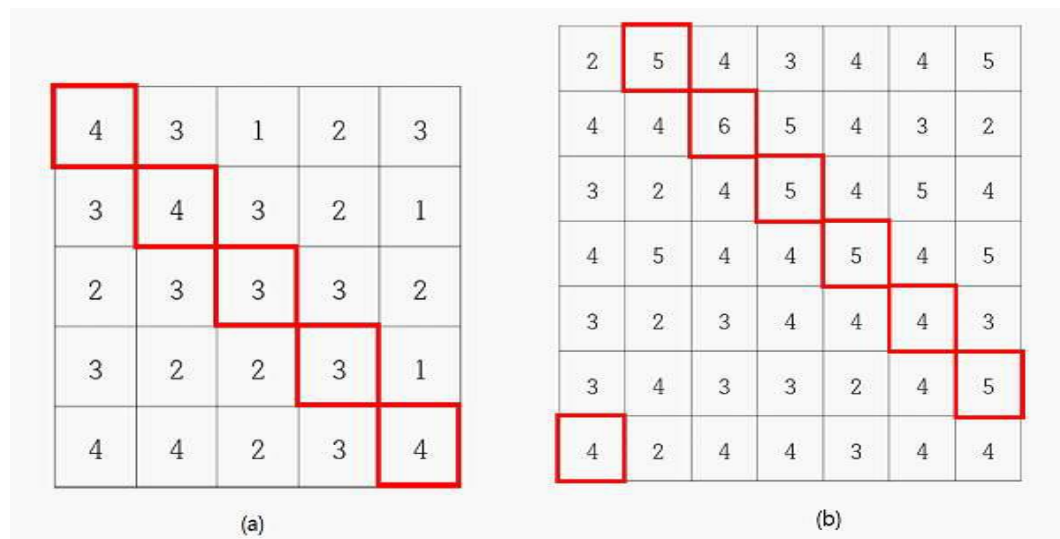
도면2



도면3



도면4



도면5

