CFG(Control Flow Graph)

* 200810048 정재근
* 200811445 이성현
* 200811414 김연준
* 200812423 김준식

Content

- CFG의 이해
  - 1. CFG의 정의
  - 2. CFG의 구성 요소
  - 3. CFG의 구성 단계
- CFG Algorithm의 종류
  - 1. Positioning Algorithms
  - 2. Routing Algorithms
- 선택한 CFG Algorithm
- Statement of Purpose
- Q&A

CFG(Control Flow Graph)의 정의

* CFG는 컴퓨터 프로그램의 코드가 취할 수 있 는 다양한 길의 가상의 표현

* CFG는 프로그램의 실행을 표현하는 그림

* CFG의 전체 과정은 그래프로 이루어진 기호 로 형성

CFG(Control Flow Graph)의 정의

* CFG는 프로그램의 모든 세부적인 부분의Control Flow를 포함

* CFG는 하나의 그림에 모든 것을 나타낼 수 있기에 보다 쉬운 분석과 컴파일러 최적화를 하는데 사용

CFG(Control Flow Graph)의 구성요소

* **Basic Block :** 곧은 선들로 만든 도형으로 표시하고 프로그램 안에서 분석을 넣은 코드의 한 부분

* **Node :** 하나 또는 그 이상의 절차적인 설명을 나타내며 코드 상태를 포함하는 기본 단위

* **Loop :** 조건이 만족될 때까지 반복하여 실행할 수 있는 명령의 집합

CFG(Control Flow Graph) 의 구성요소

* edge : 두 Node 사이의 흐름을 나타내는 연결선
* back edge : 한 정점에 이르렀을 때 벗어나는 과정에서 사용하는 edge
* critical edge : 첫 block에서 떠나거나, 도착 block를 들어가는 edge가 아닌 edge
* abnormal edge : 알 수 없는 도착점을 향한 edge로 최적화가 되지 못하게 하는 edge
* impossible edge : 그래프에 단지 block의 내용을 지키려고 추가되는 edge

CFG(Control Flow Graph) 의 구성단계

* ① 저장된 매개 언어에서 **Control Flow Graph** 생성에 필요한 함수 정보만을 추출하여 통합 저장 파일에 저장하는 단계

* ② **Control Flow Graph**의 생성에 필요한 입력 그래프의 노드 및 간선들의 정보들을 매개 언어로부터 탐색하여 **Control Flow Graph** 의 정보 모형에 저장하는 단계

* ③ 레벨 알고리즘에 의해 노드의 레벨을 정하고, 간선의 종류를 식별 하는 단계

* ④ 순서 알고리즘에 의해 각 레벨에 있는 노드들의 순서를 정하는 단계

* ⑤ 위치 알고리즘에 의해 노드와 간선들의 절대적인 **X, Y** 좌표를 생성하는 단계

* ⑥ **Control Flow Graph**와 **Call Tree**를 캔버스상에 출력하는 단계

CFG Algorithm의 종류

(Positioning Algorithm)

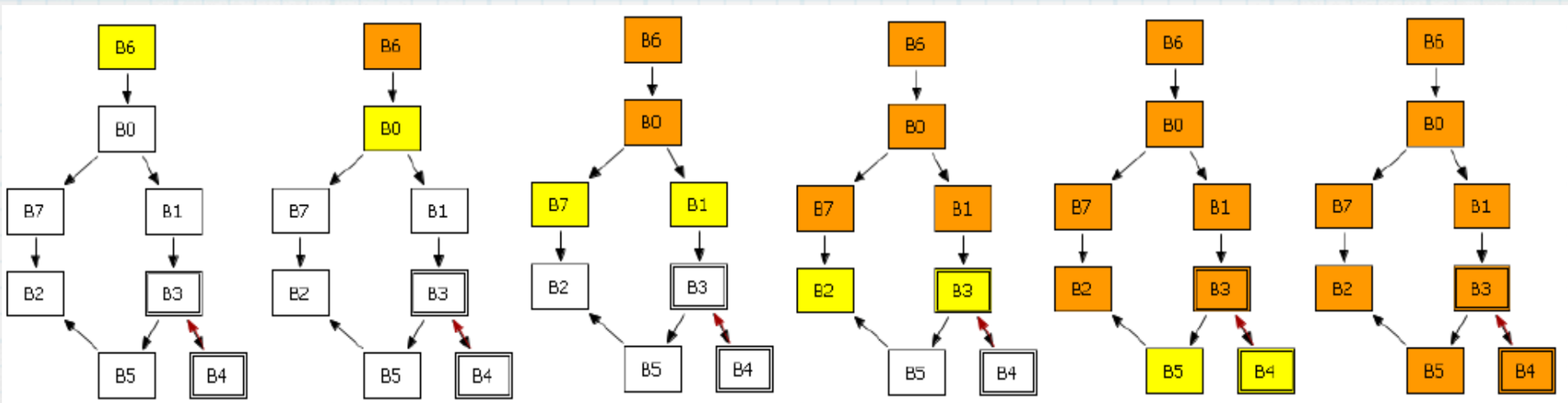
Positioning Algorithm은 구체적인 값의 모든 **node**들이 **x**좌표와 **y**좌표에 놓여있으며 **edge**보다는 **block** 중심의 알고리즘으로 아래와 같은 종류가 있다.

BFS Positioning Algorithm
Loop Positioning Algorithm
Hierarchical Positioning Algorithm

CFG Algorithm의 종류

1. BFS Positioning Algorithm

BFS Positioning Algorithm(BFS 알고리즘)은
트리 구조로 원이 없어 포괄적인 알고리즘이며 총 그래프의 길이가 길지 않아 node의 숫자가 많더라도 좋은 시각화를 가지고 있다.
하지만 사용자가 그래프에서 control flow의 정보를 읽기 힘든 가독성의 단점을 가지고 있다.
아래 그림과 같이 BFS(Breadth-first search)는 넓이 우선 탐색으로 시작한 후 모든 인접한 정점을 우선 방문하는 방법이며 모든 정점을 방문할 때까지 계속 실행된다.



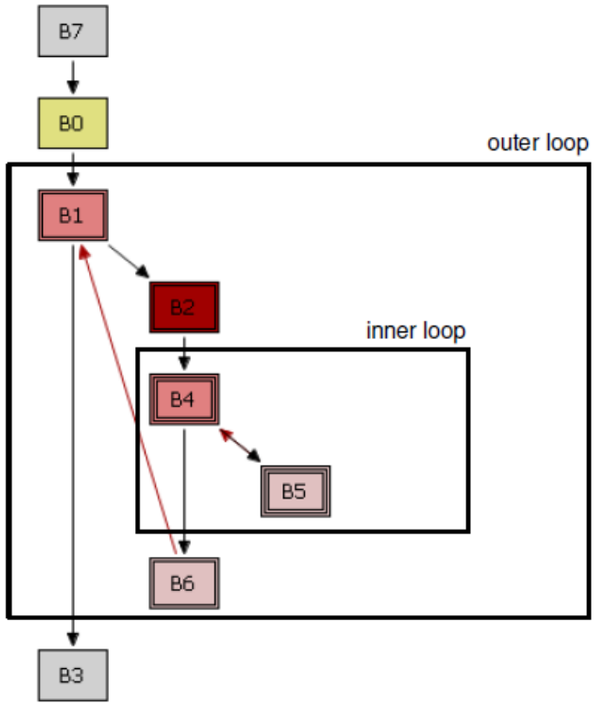
CFG Algorithm의 종류

2. Loop Positioning Algorithm

* **Loop Positioning Algorithm**은 루프를 세분화하고 소스코드와 비슷한 **Block**들을 표현하는 것이다.

* 이 알고리즘 그룹들은 **sub graphs** 와 함께 루프된다 가장 바깥쪽 범위는 루프 **header**로써 **root** 노드와 함께 취급된다

* 위 그림에서 프로그램의 **block**들 루프 중에서 바깥쪽은 **outer loop**라는 네모로 묶었고 안쪽 **block**들의 루프는 **inner loop**라는 네모로 묶어놓고 있다.

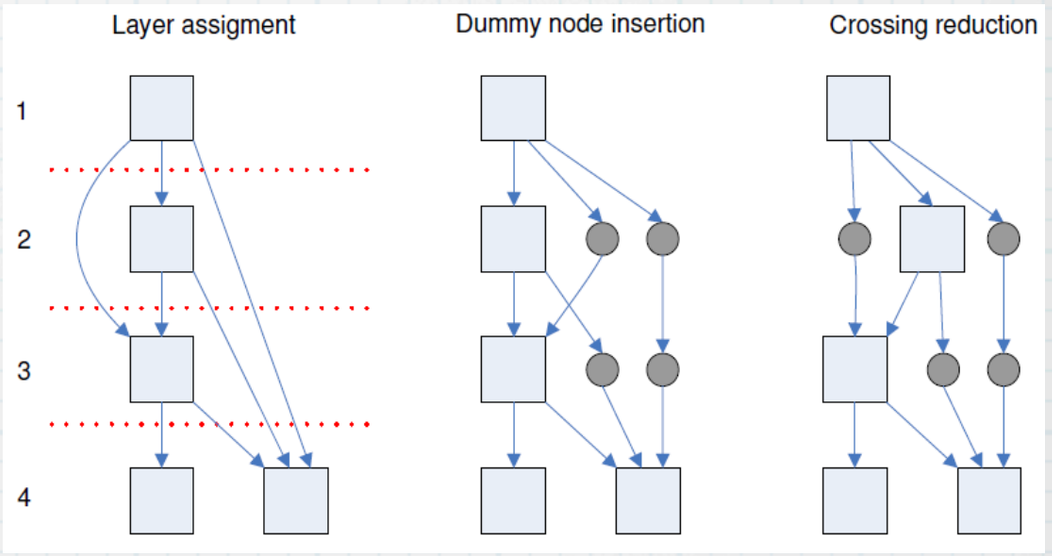


CFG Algorithm의 종류

3. Hierarchical Positioning Algorithm

Hierarchical Positioning Algorithm은 그래프에서 위치를 계산하는데 흔하게 사용되어지는 알고리즘이다. 알고리즘 내에 구체적인 정보를 사용하지 않음에도 불구하고 복잡한 교차를 줄이고 다른 알고리즘과 다르게 최적화가 되어 있기에 이 알고리즘은 사용자가 읽기에 편하다. 하지만 같은 **loop**의 **node**들이 같이 뭉쳐있지 않아 **loop**를 사용자들이 보기에 매우 어렵다.

오른쪽의 그림은 이 알고리즘이 정리하는 단계이며 흐름에 맞게 **node**들과 **edge**들을 넣고, **dummy node**를 삽입한 후에 마지막에 보기 좋게 교차 되어있는 **edge**들을 정리하는 단계를 그린 그림이다.



CFG Algorithm의 종류

(Routing Algorithm)

Positioning Algorithm은 구체적인 값
의 모든 node들이 x좌표와 y좌표에
놓여있으며 edge보다는 block 중심
의 알고리즘

BFS Positioning Algorithm
Loop Positioning Algorithm
Hierarchical Positioning Algorithm

Routing Algorithms

(Bezier Routing)

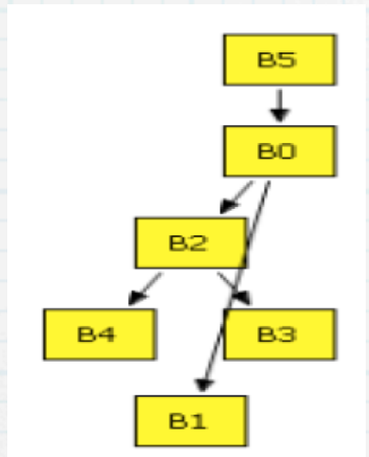
* 알고리즘의 기본발상은 사람이 종이에 적는 것에서 시작되었다.

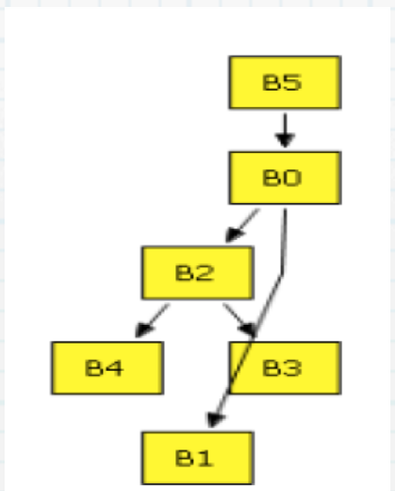
* 직선이 서로 만나지 않게 그리고, 불가피한상황. 예를 들어 장애물을 피하기 위해서는 곡선으로 그린다고 하면, 간단하지만 컴퓨터에게는 꽤나 힘들고 복잡한 작업이 된다.

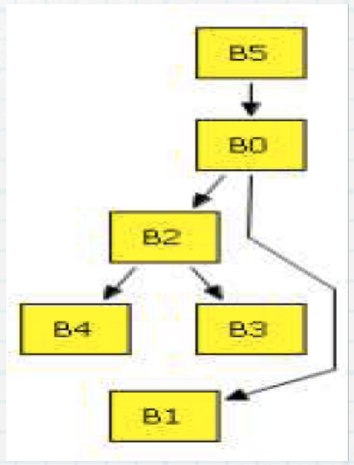
* 따라서 한단계 한단계 알고리즘을 구성해주어야한다.

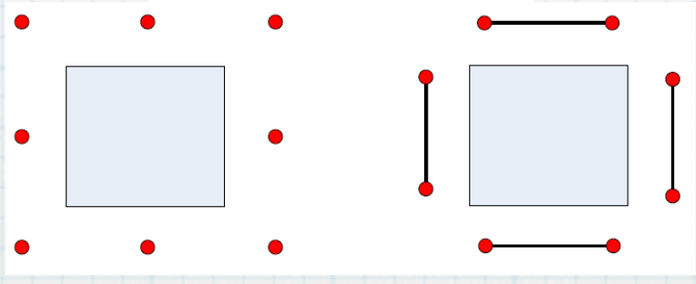
Routing Algorithms

(Bezier Routing)





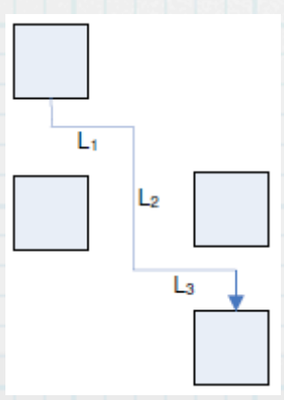


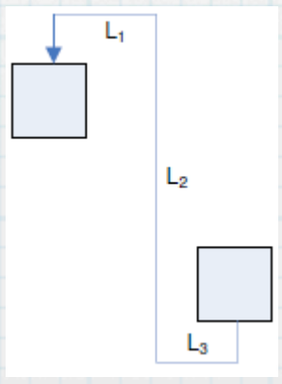


Routing Algorithms
(Manhatten Routing)

Manhatten Routing은 모든 **edge**가 직각으로 이루어져있으며,
노드로 들어가는 정보는 상단으로 들어가며, 노드에서 나가는 정보는 노드의 하단에서 나간다. 모호함을 피하기위해 노드에서 나오는 **edge**는 한,두개정도로 나오나 , 노드로 들어가는 정보는 오직하나뿐이다.

Manhattan routing은 대개 작고 대칭적이고 노드들이 직각으로 이루어진 그래프를 그릴때 수월하다





선택한 CFG Algorithm

Positioning Algorithm
- Loop positioning Algorithm

* loop들이 함께 뭉쳐져 있고, loop들을 건너뛰는 것을 사용하지 않고 break 구조를 포함하고 있으며 코드와 비슷한 장점으로 사용자가 보기에 좋다.

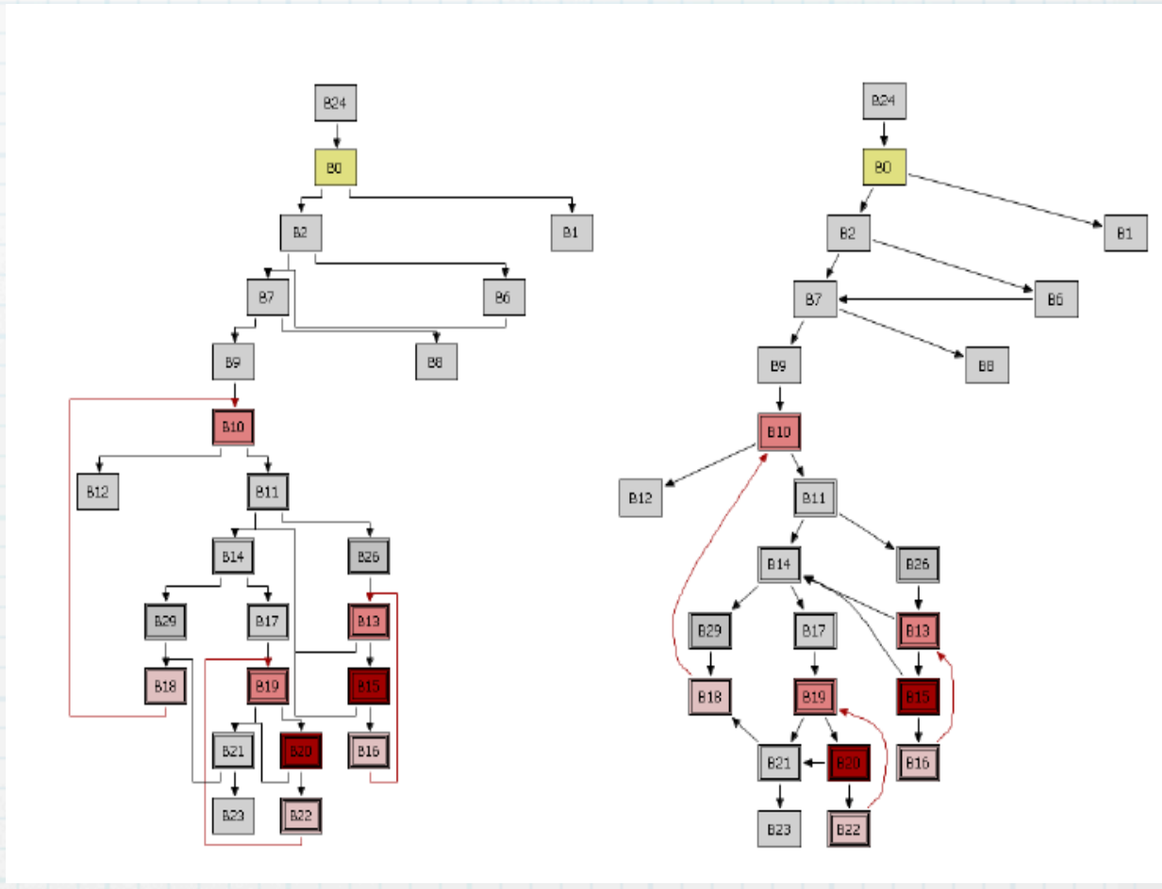
* Loop Positioning Algorithm은 다른 Positioning 알고리즘보다 긴 그래프로 그려져 내용이 복잡하더라도 편하고 깔끔하게 표현할 수 있다.

선택한 CFG Algorithm

Routing Algorithm -Bezier Routing

* 개발자가 손으로 그리는 초안과 가장 형태가 비슷하기 때문에

* Bezier Routing은 Manhattan Routing과는다르게 곡선을 사용하는 그래프로 가독성이 좋다.



Statement of Purpose

* 프로그램의 불필요한 부분이 있는지 아닌지를 발견하는 것이 주요 목적이다.
* 프로그램 실행에서 하나의 **node**를 넘어서 이동할 수 없는 무한한 **loop**들과 같은 고립된 프로그램을 도와준다.
* 프로그램과 그래프의 상관성을 만드는데 도움을 준다.
* **CFG**는 많은 방법으로 표현되어지며 **CFG**를 만드는 사람에 따라 다르게 나타난다.

Q&A