알고리즘 HW2 Graph Pattern Matching Challenge Report

2015-14257 김유진

2015-15711 민준기

1. 프로그램 실행 환경 및 실행 방법

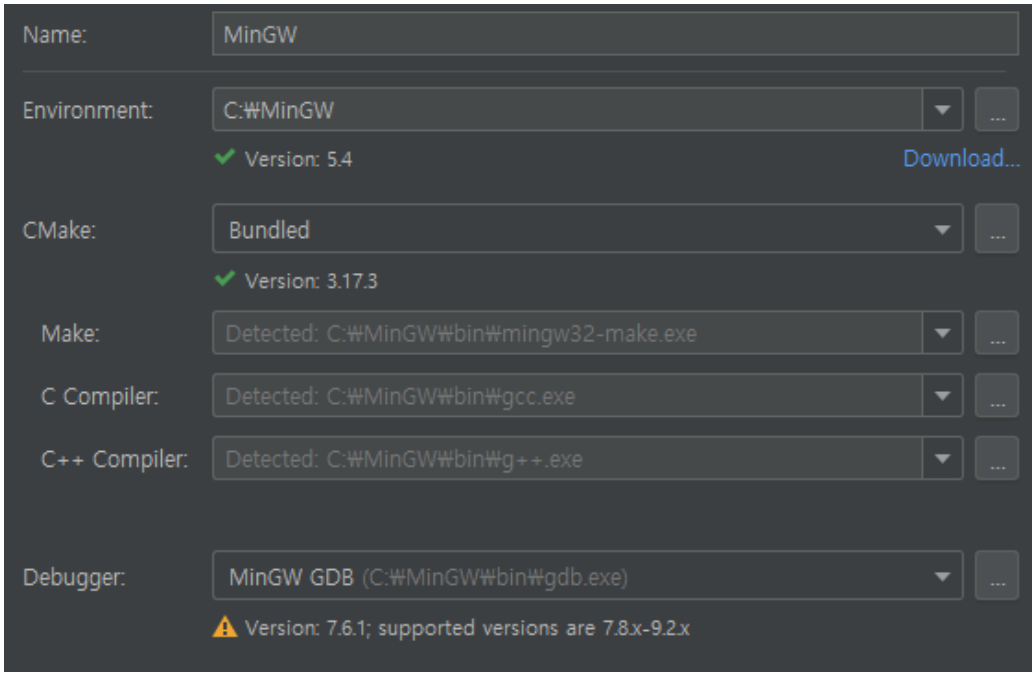
C++를 사용하여 구현하였으며, CMake version은 3.20.3, MinGW는 9.2.0.에서 활용하였다.

```
PS C:\Users\eugene> cmake --version
cmake version 3.20.3

CMake suite maintained and supported by Kitware (kitware.com/cmake).
PS C:\Users\eugene> gcc --version
gcc.exe (MinGW.org GCC Build-2) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

PS C:\Users\eugene> g++ --version
g++.exe (MinGW.org GCC Build-2) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

MinGW version 5.4, CMake version 3.17.3, debugger version 7.6.1에서도 실행을 확인 했다.



개발 IDE는 CLion을 사용했다.

실행방법은 크게 두 방법으로 나눌 수 있다.

1) Terminal에서 프로그램이 존재하는 폴더까지 간 후 다음 명령어를 차례로 실행한다.

mkdir build

cd build

cmake ..

make

./main/program <igraph file> <query file> <cs file>

예시) ./main/program ../data/test1.igraph ../query/test1_n1.igraph ../candidate_set/test1_n1.cs

예시에서 볼 수 있듯이 각 input 파일들이 다른 폴더에 있기에 이를 나타내 준다.

실행 결과 예시는 아래와 같다.

```
PS D:\2021 SNU\Algorithm\Graph-Pattern-Matching-Challenge-master\build> ./main/program ../data/lcc_hprd.igraph ../query/
lcc_hprd_n1.igraph ../candidate_set/lcc_hprd_n1.cs
t 50
a 1047 2330 1293 161 303 1126 541 1106 5008 3088 1469 221 5672 5671 5670 5669 211 4723 1684 1376 110 687 266 86 131 1168
 1379 2806 2805 684 685 4455 4468 123 5138 0 5137 1517 4700 404 160 831 280 1932 1929 179 8785 69 43 1995
a 1047 2330 1293 161 303 1126 541 1106 5008 3088 1469 221 5672 5671 5670 5669 211 4723 1684 1376 110 687 266 937 131 116
8 1379 2806 2805 684 685 4455 4468 123 5138 0 5137 1517 4700 404 160 831 280 1932 1929 179 8785 69 43 1995
a 1047 2330 1293 161 303 1126 541 1106 5008 3088 1469 221 5672 5671 5670 5669 211 4723 1684 1376 110 687 266 2599 131 11
68 1379 2806 2805 684 685 4455 4468 123 5138 0 5137 1517 4700 404 160 831 280 1932 1929 179 8785 69 43 1995
a 1047 2330 1293 161 303 1126 541 1106 5008 3088 1469 221 5672 5671 5670 5669 211 4723 1684 1376 110 687 111 730 131 116
8 1379 2806 2805 684 685 4455 4468 123 5138 0 5137 1517 4700 404 160 831 280 1932 1929 179 8785 69 43 1995
a 1047 2330 1293 161 303 1126 541 1106 5008 3088 1469 221 5672 5671 5670 5669 211 4723 1684 1376 110 687 111 2599 131 11
68 1379 2806 2805 684 685 4455 4468 123 5138 0 5137 1517 4700 404 160 831 280 1932 1929 179 8785 69 43 1995
a 937 2330 1293 161 303 1126 541 1106 5008 3088 1469 221 5672 5671 5670 5669 211 4723 1684 1376 110 687 266 86 131 1168
1379 2806 2805 684 685 4455 4468 123 5138 0 5137 1517 4700 404 160 831 280 1932 1929 179 8785 69 43 1995
a 937 2330 1293 161 303 1126 541 1106 5008 3088 1469 221 5672 5671 5670 5669 211 4723 1684 1376 110 687 266 2599 131 116
8 1379 2806 2805 684 685 4455 4468 123 5138 0 5137 1517 4700 404 160 831 280 1932 1929 179 8785 69 43 1995
a 937 2330 1293 161 303 1126 541 1106 5008 3088 1469 221 5672 5671 5670 5669 211 4723 1684 1376 110 687 111 730 131 1168
 1379 2806 2805 684 685 4455 4468 123 5138 0 5137 1517 4700 404 160 831 280 1932 1929 179 8785 69 43 1995
a 937 2330 1293 161 303 1126 541 1106 5008 3088 1469 221 5672 5671 5670 5669 211 4723 1684 1376 110 687 111 2599 131 116
8 1379 2806 2805 684 685 4455 4468 123 5138 0 5137 1517 4700 404 160 831 280 1932 1929 179 8785 69 43 1995
```

이 output은 참고용으로 answer_hprd_n1.txt 파일에도 기록해두었다.

2) termimal에서 프로그램이 있는 폴더까지 간 뒤 다음 명령어를 수행하면 된다.

→cmake-build-debug\main\program <igraph file> <query file> <cs file>

(e.g. cmake-build-debug\main\progam data\lcc_hprd.igraph query\lcc_hrpd_n1.igraph candidate_set\lcc_human_n3.cs)

```
C:\Users\als15\Documents\GitHub\Graph-Pattern-Matching-Challenge>cmake-build-debug\main\program data\lcc_hprd.igraph query\lcc_hprd_n1.igraph candidate_set\lcc_hprd_n1.cs
```

여기서 처음 명령어를 이렇게 한 이유는 debug를 하여 프로그램을 생성하면 program.exe가 cmake-build-debug\main 폴더에 생성되기 때문이다. program.exe만 terminal을 통해 실행시켜주면 정상적으로 잘 실행이 된다.

2. 코드 설명

1) Matching order

이 코드에서 matching order는 기본적으로 candidate size order를 따랐다. 가장 먼저 root가 될 vertex를 탐색하고 이렇게 찾은 root부터 시작하여 BFS search를 활용하여 DAG를 구성한다.

$$\text{root } r \leftarrow \underset{u \in V(q)}{\operatorname{argmin}} \frac{|C_{\text{ini}}(u)|}{\deg_q(u)}.$$

이렇게 구성된 DAG를 활용할 때 parent vertex를 사용하는 경우를 위해 inverse DAG도 구성해둔다. 둘 모두 2D vector 형태로 구성하였다.

그 다음 DAG를 따라서 현재 embedding을 기준으로 extendable vertex를 찾는데, 이 때 가능한 extendable vertex u에 matching 될 수 있는 후보 v들이 제일 적은 vertex 부터 search를 시작하는 candidate size order로 구현하였다. 이를 위해 다음 후보v를 정할 때 candidate vertex 중에서 candidate size가 가장 작은 vertex를 탐색하는 과정을 거쳤다.

2) Backtracking

주어진 candidate set 중에 적합한 vertex에 mapping하는 과정을 반복하고 이 과정에서 mapping이 이루어질 때마다 extendable vertex와 extendable candidates를 update 해준다. vertex mapping은 backtracking 과정으로 recur이라는 이름의 function을 통해 recursive하게 이루어지며 extendable vertex와 extendable candidates를 update하는 과정은 get_new_extendable_pair라는 이름의 function에서 이루어진다.

get_new_extendable_pair function에서는 앞서 언급했듯이 vertex mapping이 완료된 뒤에 extendable vertex와 extendable candidates를 update를 해주는 과정을 수행한다. mapping이 완료된 vertex는 더 이상 확인을 할 필요가 없기 때문에 mapping이 완료되었다고 표시를 해 준 뒤 이 vertex와 대응되는 candidates가 있는 extendable pair를 제거한다. 이 과정에서 변화는 이번에 mapping된 vertex 뿐이므로 새로운 extendable vertex는 이번에 mapping된 vertex의 child들만 확인해주면 된다. 이렇게 확인하여 extendable vertex들과 이에 대응되는 candidates를 구한 뒤 이를 pair로 묶어서 extendable pair를 update해주고 update된 pair를 return해준다.

recur function에서는 이렇게 update된 extendable pair에서 다음으로 matching을 진행할 vertex를 선택한다. 이 때, extendable pair가 비어있다면 mapping이 완료되었거나 중간에 더 이상 mapping을 진행할 수 없는 상황이기 때문에 mapping이 완료되었으면 결과를 출력하고 그렇지 않으면 그냥 return을 해주어 back track을 해주면 된다. 만약 extendable pair가 비어있지 않다면 다음으로 mapping을 진행할 vertex를 선택한다. 이는 extendable pair에서 extendable candidates의 size가 가장 작은 pair를 선택한 뒤 이 과정을 recursive하게 진행하면 embedding들을 찾을 수 있다.