Graph Pattern Matching Challenge Report

고지형 2018-16165 , 구본류 2020-10856

1. 알고리즘 설명

- query vertex 탐색 순서

query vertex의 방문 순서는 search_seq에 기록한다.

1. candidate set의 size가 3 이하인 경우 그 set는 반드시 초기에 방문하도록 사전에 검색하여 search_seq에 삽입한다.
2. 이후부터는 남은 query vertex들은, 이전까지 search_seq에 삽입되었던 vertex들과의 connection의 개수를 비교해 그 개수가 가장 큰 vertex가 search_seq에 다음으로 삽입된다.
3. 만약 connection의 개수가 같다면, candidate set의 size가 작은 query를 먼저 삽입한다.

- 한 candidate set 안에서 원소의 탐색 순서

candidate set안의 모든 vertex에 weight를 매겨, 그 weight가 큰 순서대로 탐색의 대상이 된다.

query vertex의 방문 순서인 search_seq를, DAG가 위상 정렬된 형태라고 생각한다.

search_seq를 역순으로 탐색하며, 각 query vertex의 candidate set 원소는 자신보다 위상이 낮은(search_seq의 뒤쪽) query vertex와 연결될 수 있는 경우의 수를 weight로 가진다. 만약 자신과 연결되어 있고 위상이 자신보다 낮은 query vertex가 없는 query vertex A가 있으면 A의 candidate set의 모든 원소의 weight는 1이 된다.

예를 들어, search_seq가 c1(a,b,c) -> c2(d,e) 라면 d, e는 최말단이므로 각각 1을 weight로 가진다. 이제 weight 계산은 c1으로 넘어가서, c1의 원소 a가 만약 data에서 d,e와 연결되어 있다면 1+1 = 2가 a의 weight가 된다. 만약 연결되어 있는 위상이 자신보다 낮은 query vertex가 2개 이상일 경우, 각각에 대한 연결의 경우의 수 끼리 곱한 것이 weight가 된다.

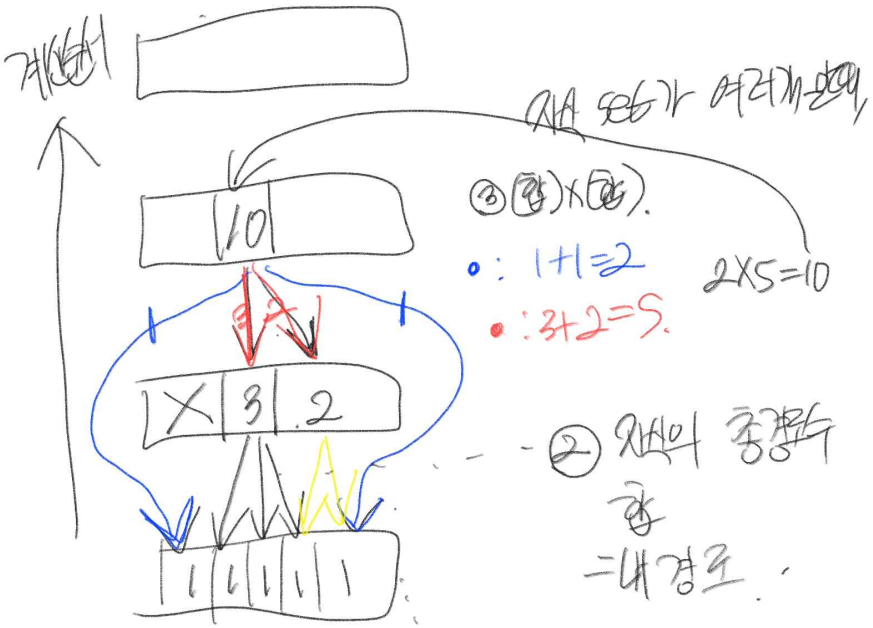
- 참고 그림1(2페이지)

2. 컴파일러 정보

gcc (Ubuntu 8.4.0-1ubuntu1~18.04) 8.4.0

3. 실행방법 및 컴파일 방법

mkdir build
cd build
cmake ..
make
./main/program <data graph file> <query graph file> <candidate set file>
checker의 경우,
g++ -v checker.cpp ./src/graph.cc -o test
./test <data graph file> <query graph file> <output file>


10
X
3
2
1
1
1
1
1
2X5=10