

CSE3081 (2반): 알고리즘 설계와 분석

[숙제 3 (version 1.0)]

담당 교수: 임 인 성

2020년 12월 14일

마감: 12월 26일 토요일 오후 8시 정각 (LATE 없음)

제출물 및 제출 방법 등: 조교가 사이버 캠퍼스에 공지할 예정입니다.

목표: (1) 알고리즘 설계 기법 중의 하나인 greedy 방법에 대한 이해도를 높이도록 한다. (2) Kruskal의 Minimum Spanning Tree 알고리즘을 구현하여 본다. (3) Disjoint-sets 자료 구조를 사용하여 프로그램의 효율을 높여본다. (4) Min heap을 실제 문제 해결에 적용하여 본다. (5) 방대한 크기의 그래프를 효율적으로 처리하여 본다.

- 이번 숙제에서 사용하는 *weighed undirected graph* $G = (V, E)$ 는 다음과 같은 형식으로 ASCII 파일에 저장 되어있다.

```
334863 925872 65536
0 53525 25038
0 71631 17666
0 98005 16296
0 148223 1717
0 209319 5450
0 268298 29553
:
```

첫 줄의 앞 두 숫자는 각각 이 그래프의 vertex의 개수 `n_vertices`와 edge의 개수 `n_edges`를 나타낸다. 이 줄의 마지막 숫자는 edge의 weight가 가질 수 있는 가장 큰 값 `MAX_WEIGHT`를 나타낸다(이때 edge의 weight는 1에서 `MAX_WEIGHT`까지의 정수 값을 가질 수 있음).

두 번째 줄부터는 `n_edges`의 개수 만큼 각 edge에 대한 (from-vertex ID, to-vertex ID, weight) 정보가 나열된다(이때 vertex ID는 0부터 `n_vertices-1`까지의 값을 가지며, 이 그래프는 *undirected graph*임을 명심할 것). 이 그래프는 *connected graph*가 아닐 수도 있으며, 여러분은 그러한 사항을 고려하여 프로그램을 작성해야 한다.

- 이번 숙제에서 제공하는 총 6개의 서로 다른 크기의 그래프들은 다음과 같은데, 이들은 모두 <https://snap.stanford.edu/data/>에서 제공하는 그래프들을 기반으로 생성하였다.

file name	n_vertices	n_edges
HW3_email-Eu-core.txt	1,005	25,571
HW3_com-dblp.ungraph.txt	317,080	1,049,866
HW3_com-amazon.ungraph.txt	334,863	925,872
HW3_com-youtube.ungraph.txt	1,134,890	2,987,624
HW3_wiki-topcats.txt	1,791,489	28,511,807
HW3_com-lj.ungraph.txt	3,997,962	34,681,189

3. 자신이 작성한 프로그램은 (i) 위의 그래프 데이터 파일들과 동일한 디렉터리에 존재하는 이름이 `commands.txt`인 파일에서 입력 그래프 파일의 이름을 읽어 들인 후 (아래의 예에서와 같이 이 파일의 첫 줄에 입력 파일의 이름이 주어짐),

```
HW3-com-dblp.ungraph.txt
HW3-com-dblp.ungraph.MST_result.txt
```

(ii) 해당 그래프에 대한 **minimum spanning tree**를 구한 후, (iii) 위 `commands.txt` 파일의 두 번째 줄에 주어진 출력 파일의 이름에 그 결과를 ASCII 파일 형식으로 저장한다. 이 결과 파일의 형식을 다음과 같은 예를 통하여 설명하면,

```
3
317024 2394950493
30 95433
26 67872
```

첫 줄에는 해당 그래프의 **connected component**의 개수가 주어지고, 다음 그 개수만큼의 줄 각각에 (number of vertices in the component, total weight of minimum spanning tree) 정보를 기술해야 한다. 만약 어떤 **connected component**가 한 개의 꼭지점으로 구성되어 있다면 (**isolated vertex**), 해당 줄에는 '1, 0' 값이 저장되어야 하고, 전체 그래프가 **connected**되어 있다면 'n_vertices, total weight of minimum spanning tree' 정보가 저장되어야 한다.

4. 이번 숙제에서는 반드시 **disjoint-sets/min heap** 자료 구조에 기반을 둔 최악의 경우의 수행 시간이 $O(|E| \log |V|)$ 인 알고리즘을 구현하여야 한다. 자신이 그렇게 구현 하였다는 것을 밝히기 위하여, 자신이 작성한 원시코드를 아래의 그림과 같이 출력한 후, 그 위에 코드 구현에 대한 설명을 기술하여 제출 기한

```

...W4WProgWGraph_generatorWGraph_generatorWGraph_generator.cpp 1
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 int main() {
6
7     char input_file_name[128] = "../../../Data/com-dblp.ungraph_tmp.txt";
8     char output_file_name[128] = "../../../Data/HW4-com-dblp.ungraph.txt";
9
10    FILE *fp_in, *fp_out;
11    int n_vertices, n_edges, v_from, v_to;
12
13    /* Find_Connected_Component(output_file_name); */
14
15    fp_in = fopen(input_file_name, "r");
16    if (!fp_in) {
17        fprintf(stderr, "^^^ Error: cannot open the file %s.\n", input_file_name);
18        exit(-1);
19    }
20
21    fscanf(fp_in, "%d %d", &n_vertices, &n_edges);
22    fprintf(stdout, "*** N_VERTICES = %d, N_EDGES = %d\n", n_vertices, n_edges);
23
24    // Find the largest vertex id.
25    int v_max = -1;
26    for (int i = 0; i < n_edges; i++) {
27        fscanf(fp_in, "%d %d", &v_from, &v_to);
28        //if (v_from >= v_to) fprintf(stdout, "*** v_from >= v_to = %d <= %d\n",
29        //    v_from, v_to);
30        if (v_from > v_max) v_max = v_from;
31        if (v_to > v_max) v_max = v_to;
32    }
33    fprintf(stdout, "... The last edge is from %d to %d\n", v_from, v_to);

```

내에 조교에게 제출하라. 자신이 사용한 graph representation 방법과 disjoint-set 구현 방법 등을 포함한 개략적인 설명이 필요하며, 왜 $O(|E| \log |V|)$ 의 시간이 소요되는 알고리즘인지를 또한 개략적으로 설명할 것.

5. 바로 위에서 기술한 코드 설명 문서의 제일 앞장에는 자신의 실험 결과를 아래와 같은 양식의 표에 요약하여 제출하라(조교는 이 내용에 따라 여러분의 프로그램을 컴파일 한 후 그 결과를 확인할 예정임).

파일 이름	작동 여부	MST weight	수행 시간(초)	$k_{scanned}$
HW3_email-Eu-core.txt	YES	49,388	0.7 (0.4)	13,239
HW3_com-dblp.ungraph.txt	YES	989,403초	12.3 (8.1)	832,391
HW3_com-amazon.ungraph.txt	YES	1,200,359	17.4 (12.8)	473,543
HW3_com-youtube.ungraph.txt	YES	12,349,057	37.2 (27.4),	1,943,543
HW3_wiki-topcats.txt	YES	243,965,012	41.9 (29.5)	11,239,518
HW3_com-lj.ungraph.txt	NO	-	-	-

- “MST weight”에는 가장 꼭지점이 많은 컴포넌트에 대한 minimum spanning tree의 결과 weight를 기술하고, 동일한 크기의 컴포넌트가 두 개 이상일 경우 각각에 대한 결과 웨이트를 기술하라.
- 전체 계산 과정 중 “edge를 weight의 크기에 따라 나열하는 계산”은 먼저 전체 edge들에 대해 weight 값을 key로 하여 min heap을 구축한 후, weight가 작은 순서로 하나씩 delete-min하는 방식을 취하라.
- “수행 시간(초)”에는 먼저 전체 수행 시간(입출력 시간을 제외한 순수 실행 시간)을 기술하라. 다음 괄호 안에는 이 전체 시간 중 바로 위에서 기술한 “min heap을 구축하는 과정”에 걸리는 시간(이론적으로 $O(|E|)$)을 가급적 정확히 측정하여 기술하라.
- 다음 “ $k_{scanned}$ ”는 minimum spanning tree의 구축이 완성될 때까지 처리한 edge의 개수를 의미함.
- 이때 “disjoint-sets 자료 구조 처리에 소요되는 계산”은 이론적으로 $O(k_{scanned} \cdot \log |V|)$ (최악의 경우 $(|E| \log |V|)$)의 시간 또는 path compression 기법을 적용할 경우 $O(k_{scanned} \cdot \alpha(|V|))$ (최악의 경우 $(|E| \cdot \alpha(|V|))$ 의 시간이 걸리다는 사실을 명심하라(여기서 $\alpha(\cdot)$ 은 inverse Ackermann function을 나타냄).
- 여러분이 측정한 실행 시간이 과연 위의 이론적인 시간 복잡도를 반영한다고 할 수 있는지 자신의 분석 결과를 명확히 기술하라.
- 또한 필요할 경우 이 테이블을 포함하는 보고서 문서에 채점 시 조교가 알아야 할 것이 있을 경우 명확히 기술하라.

• [주의]

1. 이번 숙제에서는 실제 minimum spanning tree에 대한 정보를 출력할 필요 없이 weight들의 총 합만 출력하면 되나, minimum spanning tree를 구축해가면서 어떻게 하면 **효율적으로** tree 정보를 저장할 지 생각해볼 것.
2. 숙제를 제출할 때 **데이터 파일은 반드시 삭제하고 보낼 것**(입출력 파일이 존재하는 디렉토리는 조교가 공지할 예정임).
3. 채점을 할 때, Visual Studio 2019 상에서 **솔루션 구성은 Release**로 그리고 **솔루션 플랫폼은 x64** 모드를 사용할 예정이며, 만약 다른 구성을 사용하여 프로그램을 작성한 것으로 인하여 컴파일이 안되거나 프로그램이 수행이 안될 경우 이는 본인의 책임임을 명심하라.
4. 숙제 제출 기간 동안 조교가 숙제 및 기타 채점 결과와 관련하여 중요한 공지 사항을 게시판에 올리거나 이메일을 보낼 수 있으니 주의하기 바람.
5. **[중요]** Copy-check 결과 copy로 판명이 날 경우 **관련된 사람 모두에 대하여** 최종 성적에 상당한 영향을 미칠 감점이 있을 예정임.
6. **[더 중요]** 프로그램을 작성하는데 있어서, 수업시간에 다룬 예제 코드나 웹에서 구한 코드와 유사한 것으로 판명이 날 경우 copy로 처리할 예정이니, 반드시 자신의 언어/방식으로 혼자서 프로그램을 작성하기 바랍니다.