CSE3081 (2반): 알고리즘 설계와 분석
HW 3: Minimum Spanning Tree 알고리즘의 구현

담당 교수: 임 인 성

2022년 12월 5일

마감: 12월 23일 금요일 오후 8시 정각 (LATE 없음)
제출물 및 제출 방법 등: 조교가 사이버 캠퍼스에 공지할 예정임.

목표: (1) 알고리즘 설계 기법 중의 하나인 greedy approach에 대한 이해도를 높이도록 한다. (2) Prim과 Kruskal의 Minimum Spanning Tree 알고리즘 각각을 구현하여 본다. (3) 자신이 구현한 각 방법의 이론적인 시간 복잡도와 실제 수행 시간과의 관계를 분석하여 본다. (4) 대용량 그래프 처리를 위한 효율적인 자료 구조 설계 및 구현에 관하여 연습하여 본다.

1. 이번 숙제에서 사용하는 *weigthed 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-verex ID`, `to-vertex ID`, `weight`) 정보가 나열된다 (이때 vertex ID는 0부터 `n_vertices`-1까지의 값을 가지며, 이 그래프는 undirected graph임을 명심할 것). 이 그래프는 connected graph가 아닐 수도 있으며, 여러분은 그러한 사항을 고려하여 프로그램을 작성해야 한다.

2. 이번 숙제에서 제공하는 총 6개의 서로 다른 크기의 그래프들은 다음과 같은데, 이들은 모두 `https://snap.stanford.edu/data/`에서 제공하는 그래프들을 기반으로 생성하였다. 서강 사이버 캠퍼스에 공지하는 링크에서 그래프들을 다운로드 받아 이 테이블의 파일명으로 변경한 후 실험에 사용하라.

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 각 프로그램은 (i) Visual Studio의 solution directory에 존재하는 이름이 `commands.txt` 인 파일의 첫 번째 줄에서 입력 그래프 파일이 존재하는 디렉터리 정보를 읽어 들여,

```
(commands.txt 파일 예)
C:\usr\local\Algorithm\HW_3\Graphs
HW3_com-dblp.ungraph.txt
HW3_com-dblp.ungraph_MST_result.txt
```

(ii) 두 번째 줄의 파일에서 입력 그래프를 읽어 들여 minimum spanning tree를 계산한 후, (iii) 위의 `commands.txt` 파일의 세 번째 줄에 주어진 출력 파일에 그 결과를 ASCII 파일 형식으로 저장한다 (**주의: 이 파일은 `commands.txt`과 동일한 원래의 solution directory에 저장할 것**). 이 결과 파일의 형식을 다음과 같은 예를 통하여 설명하면,

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

첫 줄에는 해당 그래프의 connected component의 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 component가 한 개의 꼭지점으로 구성되어 있다면 (isolated vertex), 해당 줄에는 '`1, 0`' 값이 저장되어야 하고, 전체 그래프가 connected되어 있다면 '`n_vertices, total weight of minimum spanning tree`' 정보가 저장되어야 한다.

4. Prim의 Minimum Spanning Tree 알고리즘을 구현할 때 시간 복잡도가 $O(|V|^2)$ 방법이 아닌 **수업 시간에 설명한 min heap에 기반을 둔 최악의 경우의 시간 복잡도가 $O(|E|\log|V|)$인 방법을 구현**하여야 한다. 또한 Kruskal의 Minimum Spanning Tree 알고리즘을 구현할 때에는 **수업 시간에 설명한 disjoint sets과 min heap에 기반을 둔 최악의 경우의 시간 복잡도가 $O(|E|\log|V|)$인 방법을 구현**하여야 한다.

- **[주의]** 이번 숙제에서는 실제 minimum spanning tree에 대한 정보를 출력할 필요 없이 weight들의 총 합만 출력하면 되나, 트리를 구축해가면서 어떻게 하면 **효과적으로** tree 정보를 저장할 지 생각해볼 것.

1. **제출물 1:** 자신이 구현한 소스 코드는 이름이 `HW_3_S20219999`인 디렉터리 아래에 이름이 각각 Prim과 Kruskal인 디렉터리에 저장하여 zip으로 압축하여 보낼 것.
   - 이때 그래프 데이터 파일과 .vs 등과 같이 Visual Studio가 컴파일 및 수행 시 생성한 파일들은 반드시 삭제하고 제출할 것.
   - 채점을 할 때, Visual Studio 2022 상에서 **솔루션 구성**은 **Release**로 그리고 **솔루션 플랫폼**은 **x64** 모드를 사용할 예정이며, **만약 다른 구성을 사용하여 프로그램을 작성한 것으로 인하여 컴파일이 안되거나 프로그램이 수행이 안될 경우 이는 본인의 책임임을 명심하라.**
2. **제출물 2:** 보고서는 이름이 `HW3_S20219999.{hwp,docx,txt,pptx}`인 파일을 위에서 명시한 `HW_3_S20219999` 디렉터리 아래에 소스 코드와 같이 저장하여 위의 zip 파일에 같이 제출하라. 즉 `HW_3_S20219999` 디렉터리 아래에는 두 개의 디렉터리와 하나의 (보고서) 파일이 저장된다.

(a) 보고서의 가장 앞에 다음의 예처럼 실험에 사용한 CPU의 속도 및 메인 메모리의 용량 등의 실험 환경을 기술하라.

```
OS: Windows 10 Education
CPU: Intel(R) Core(TM) i9-10910 @ 3.60GHz 3.60GHz
RAM: 32.00GB
Compiler: Visual Studio 2022 Release Mode
```

(b) 위에서 요구한 알고리즘들을 올바르게 구현하였다는 것을 밝히기 위하여, 보고서에 자신이 작성한 원시코드의 주요 부분을 아래의 Figure 1과 같이 출력한 후 그 위에 코드 구현에 대한 설명을 기술하라. 특히 그러한 시간 복잡도를 달성하는데 필요한 핵심 부분에 대한 **자신만의 설명**이 있어야 한다.

(c) 보고서에 Prim과 Kruskal 각 방법에 대한 자신의 실험 결과를 아래와 같은 양식의 표에 요약하여 제출하라. 조교는 이 내용에 따라 여러분의 프로그램을 컴파일 한 후 그 결과를 확인할 예정이므로 각 방법에 대한 구현 여부 및 해당 데이터 처리 여부를 명확히 밝힐 것.

파일 이름	작동 여부	MST weight	수행 시간(초)	$k_{scanned}$
HW3_email-Eu-core.txt	YES	49,388	0.7	13,239
HW3_com-dblp.ungraph.txt	YES	989,403	12.3	832,391
HW3_com-amazon.ungraph.txt	YES	1,200,359	17.4	473,543
HW3_com-youtube.ungraph.txt	YES	12,349,057	37.2	1,943,543
HW3_wiki-topcats.txt	YES	243,965,012	41.9	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.
- "수행 시간(초)"에는 파일 입출력 시간을 제외한 minimum spanning tree 구축에 소요된 순수한 수행 시간만 측정하여 기술하라.
- 다음 "$k_{scanned}$"는 Kruskal 방법의 결과 테이블에만 기술하되, minimum spanning tree의 구축이 완성될 때까지 처리한 edge의 개수를 의미함. Prim 방법 결과에는 이 값을 기술할 필요가 없음.

(d) 이어서 여러분이 측정한 수행 시간이 과연 위의 이론적인 시간 복잡도를 반영한다고 할 수 있는지 자신의 분석 결과를 명확히 기술하라.

(e) 또한, 필요할 경우 채점 시 조교가 알아야 할 것이 있을 경우 명확히 기술하라.

3. 기타

- 숙제 제출 기간 동안 조교가 숙제 및 기타 채점 결과와 관련하여 중요한 공지 사항을 서강 사이버 캠퍼스에 올리거나 이메일을 보낼 수 있으니 주의하기 바람.
- **[중요]** Copy-check 결과 copy로 판명이 날 경우 **관련된 사람 모두에 대하여** 최종 성적에 상당한 영향을 미칠 감점이 있을 예정임.
- **[더 중요]** 프로그램을 작성하는데 있어서, 수업시간에 다룬 예제 코드나 웹에서 구한 코드와 유사한 것으로 판명이 날 경우 copy로 처리할 예정이니, 반드시 자신의 방식으로 혼자서 프로그램을 작성하기 바랍니다.
- **[더더 중요]** 기말고사와 숙제 채점 결과를 준비가 될 때마다 수시로 서강 사이버 캠퍼스에 공지할 예정이므로 성적 제출 마감 시간 전까지 확인하기 바랍니다. 모든 수강생들에게 공정을 기하기 위하여, 성적에 관한한 개인적인 사정을 고려하기가 매우 어려움을 이해하기 바랍니다. 모두 수고 많았습니다.

...W4₩Prog₩Graph_generator₩Graph_generator₩Graph_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     }
```

Figure 1: 원시 코드 출력 예