CSE3081 (2반): 알고리즘 설계와 분석

HW 4 (Ver. 1.0)

담당 교수: 임 인 성

2021년 12월 3일

마감: 12월 23일 목요일 오후 8시 정각 (**LATE 없음**)
제출물, 제출 방법, LATE 처리 방법 등: 조교가 과목 게시판에 공고할 예정임.

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) 방대한 크기의 그래프를 효율적으로 처리하여 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, 여러분의 프로그램은 그러한 사항을 고려하여 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.

File name	n_vertices	n_edges
HW4_email-Eu-core.txt	1,005	25,571
HW4_com-dblp.ungraph.txt	317,080	1,049,866
HW4_com-amazon.ungraph.txt	334,863	925,872
HW4_com-youtube.ungraph.txt	1,134,890	2,987,624
HW4_wiki-topcats.txt	1,791,489	28,511,807
HW4_com-lj.ungraph.txt	3,997,962	34,681,189

3. (i) 먼저 자신의 메인 함수 파일이 있는 디렉터리에 `Graphs_HW4`라는 이름의 디렉터리를 만든 후 위의 그래프 데이터 파일을 복사하라. 자신이 작성한 프로그램은 (i) 위의 `Graphs_HW4` 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),

   ```
   HW4_com-dblp.ungraph.txt
   HW4_com-dblp.ungraph_MST_result.txt
   ```

   (ii) 해당 그래프에 대한 minimum spanning tree를 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 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`"이 저장되어야 하고, 전체 그래프가 connected되어 있다면 "`n_vertices  total_weight_of_MST`"가 저장되어야 한다.

4. 이번 숙제에서는 **반드시 disjoint-sets 자료 구조에 기반을 둔 최악의 경우의 수행 시간이 $O(|E|\log|V|)$ 인 알고리즘을 구현**하여야 한다. **또한 전체 계산 과정 중 "edge를 weight의 크기에 따라 나열하는 계산" 은 반드시 min heap을 사용하는 방법 ($O(|E| + k_{scanned} \cdot \log|V|$ 시간)을 사용**하라.

5. **자신의 구현 내용이 수업 시간에 다룬 코드와 유사성이 있을 경우 기계적으로 COPY CHECK에 걸릴 수 있으므로, 그러한 일이 발생하지 않도록 주의하라.**

6. 자신의 구현 내용을 명확히 밝히기 위하여, 자신이 작성한 원시코드의 핵심 부분을 다음 쪽의 그림과 같이 출력한 후 자신의 코드 구현에 대한 간략한 설명을 기술하여, 파일로 스캔 후 보고서에 첨부하여 제출하라. 자신이 사용한 graph representation 방법과 disjoint-set 구현 방법 등을 포함한 전반적인 설명이 필요하며, 왜 $O(|E|\log|V|)$의 시간이 소요되는 알고리즘인지를 간략히 분석할 것.

7. 바로 위에서 기술한 코드 설명 문서의 **제일 앞장**에는 자신의 실험 결과를 (자신이 사용한 CPU와 Main Memory의 성능과 함께) 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)	$k_{scanned}$
`HW4_email-Eu-core.txt`	YES	49,388	0.7	13,239
`HW4_com-dblp.ungraph.txt`	YES	989,403	12.3	832,391
`HW4_com-amazon.ungraph.txt`	YES	1,200,359	17.4	473,543
`HW4_com-youtube.ungraph.txt`	YES	12,349,057	37.2	1,943,543
`HW4_wiki-topcats.txt`	YES	243,965,012	41.9	11,239,518
`HW4_com-lj.ungraph.txt`	NO	–	–	–

   - "MST weight"에는 가장 꼭지점이 많은 컴포넌트에 대한 minimum spanning tree의 결과 weight를 기술하고, 그런 컴포넌트가 두 개 이상일 경우 각각에 대한 결과 웨이트를 기술하라.
   - "수행 시간 (초)"에는 MST 계산에 소요된 전체 수행 시간 (입출력 시간을 제외한 순수 실행 시간)을 기술하라.

|...W4₩Prog₩Graph_generator₩Graph_generator₩Graph_generator.cpp 1

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {

    char input_file_name[128] = "../../../Data/com-dblp.ungraph_tmp.txt";
    char output_file_name[128] = "../../../Data/HW4_com-dblp.ungraph.txt";

    FILE *fp_in, *fp_out;
    int n_vertices, n_edges, v_from, v_to;

    /* Find_Connected_Component(output_file_name); */

    fp_in = fopen(input_file_name, "r");
    if (!fp_in) {
        fprintf(stderr, "^^^ Error: cannot open the file %s.₩n", input_file_name);
        exit(-1);
    }

    fscanf(fp_in, "%d %d", &n_vertices, &n_edges);
    fprintf(stdout, "*** N_VERTICES = %d, N_EDGES = %d₩n", n_vertices, n_edges);

    // Find the largest vertex id.
    int v_max = -1;
    for (int i = 0; i < n_edges; i++) {
        fscanf(fp_in, "%d %d", &v_from, &v_to);
        //if (v_from >= v_to) fprintf(stdout, "*** v_from >= v_to = %d <= %d₩n",
        //  v_from, v_to);
        if (v_from > v_max) v_max = v_from;
        if (v_to > v_max) v_max = v_to;
    }
    fprintf(stdout, "*** The last edge is from %d to %d ₩n", v_from, v_to);
```

- 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 "edge를 weight의 크기에 따라 나열하는 계산"과 "disjoint-sets 자료 구조 처리에 소요되는 계산"에 대한 이론적인 시간 복잡도를 반영한다고 할 수 있는지 자신의 생각을 밝혀라.
- 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 총합만 출력하면 되나, minimum spanning tree를 구축해가면서 어떻게 하면 **효율적으로** tree 정보 (연결 정보 포함)를 저장할 지 생각해볼 것.
2. 온라인으로 숙제를 제출할 때 `Graphs_HW4` **디렉터리의 그래프 데이터는 삭제하고 보낼 것**.
3. 채점을 하는데 있어 Visual Studio 2019 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.**
4. 숙제 제출 기간 동안 조교가 숙제 및 기타 채점 결과와 관련하여 중요한 공지 사항을 서강 캠퍼스에 공지하거나 이메일을 보낼 수 있으니 주지하기 바람.
5. 시간적인 제약으로 인하여 **이번 숙제에는 LATE 없으며**, 제출 화일에서 바이러스 발견 시 **본인 점수 X (-1)**이고, 다른 사람의 숙제를 복사할 경우 **관련된 사람 모두에 대하여 만점 X (-10)**임.