**CSE3081 알고리즘설계와분석 HW4 보고서**

컴퓨터공학과 2학년 20181662 이건영

1. **실험 환경**

**Operation System** : Microsoft Windows 10 Home (64-bit)

**Compiler** : Microsoft Visual Studio Community 2019

**CPU** : Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.80 GHz

**RAM** : 8.00GB

1. **실험 결과**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 파일 이름 | 작동 여부 | MST weight | 수행 시간(초) | Kscanned |
| HW3\_email-Eu-core.txt | YES | 3110161 | 0.039 | 25571 |
| HW3\_com-dblp.ungraph.txt | YES | 2747895457 | 1.096 | 1049834 |
| HW3\_com-amazon.ungraph.txt | YES | 2729670156 | 1.086 | 925855 |
| HW3\_com-youtube.ungraph.txt | YES | 14578691475 | 3.053 | 2987623 |
| HW3\_wiki-topcats.txt | YES | 5351181035 | 32.361 | 28415895 |
| HW3\_com-lj.ungraph.txt | YES | 28308045762 | 42.101 | 34681165 |

1. **구현 내용**

핵심 함수들은 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. void insertMinHeap(edge newEdge);

edge 타입의 변수를 Min Heap에 새롭게 추가한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Heap은 기본적으로 배열로 구현하였다. Heap의 현재 크기는 sizeH 변수에 저장하였으며, 최대 크기는 Edge의 개수와 같다. 데이터를 Heap에 삽입하는 방법은 Heap의 다음 자리에 새로운 원소를 넣은 후 부모 노드의 값과 비교하여 더 작으면 새로운 노드가 위로 올라가는 방식으로 구현하였다.

1. edge deleteMinHeap();

Min Heap에서 가장 작은 원소를 return 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Heap의 root 원소를 return한 후, 가장 마지막 원소를 temp로 잡고 temp의 적절한 위치를 찾아서 지정되도록 구현하였다.

1. void Find(int i);

Find 함수는 해당 노드의 최상위 부모 노드의 번호를 찾는 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

간단한 코드에서 알 수 있듯 인자로 받은 i의 부모 배열 값을 계속 찾아가다, 음수가 나오면 해당 i값을 return하는 방식이다. 아래의 Union 함수에서 더 자세히 설명하겠다.

1. void Union(int I, int j, int weight);

Union 함수는 전달받은 두 인자 i와 j를 같은 disjoint-set으로 만드는 역할을 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 각각의 parent 값을 더해 temp값을 만든다. parent의 초기값은 -1이다. 그 후 parent 값이 더욱 작은 쪽을 temp로, 큰 쪽은 작은 쪽의 index 값으로 바꾼다. 그 결과 한 쪽에는 음수가, 반대쪽에는 양수가 저장된다. 이 때 음수값의 절댓값은 해당 set에 포함된 vertex의 개수가 된다. 양수값은 해당 노드의 바로 위 부모 노드를 의미한다.

다음으로 weight를 처리한다. weightSum 배열은 초기값으로 0을 가지고 있다. 만약 부모 노드가 될 노드가 정해지면 해당 노드의 index 값을 토대로 weightSum[index]의 값을 기존 값 + 전달받은 노드의 weight값 + 자식 노드가 될 노드의 weightSum 값으로 대체한다. 결과적으로 weightSum[index]는 최종 부모 노드가 된 노드의 weight의 합을 가지게 된다.

1. void kruskal();

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

kruskal 함수는 최종적으로 MST를 구성하기 위한 함수이다. 중심이 되는 반복문의 조건은 Heap의 내용물이 있으면서, 현재 선택된 간선의 수(count)가 numV(vertex의 수) – 1보다 작을 경우이다. 즉, 모든 edge를 탐색하거나, 선택된 edge의 수가 numV – 1이 되면 kruskal 알고리즘이 종료된다.

우선 Min Heap에서 하나의 원소를 추출한다. 그 원소의 양쪽 끝 점의 index를 가지고 Find 함수를 실행한다. 그 결과 두 노드의 최상위 부모노드가 u와 v에 저장된다. 만약 두 값이 다르다면, 두 노드는 다른 set에 포함되었다는 의미이므로 Union을 진행한다. 이 때 탐색한 노드의 수(실질적으로 Heap에서 추출한 edge의 수)는 Kscanned 변수에 저장된다.

반복문이 종료되면 parent값이 음수인 개수를 세서 출력한다. 그 개수가 곧 disjoint-set의 개수이기 때문이다. 이후 parent 값이 음수인 index들의 포함된 vertex의 수와 weightSum의 값을 출력한다. 이후 프로그램은 종료된다.

우선 프로그램 초기에 Heap에 edge들을 삽입하는 과정은 O(log E) 만큼을 E번 실행하므로 O(E log E)의 시간복잡도를 지닌다. 이후 edge의 개수만큼(정확히는 Kscanned 만큼) deleteMinHeap 함수를 실행하고(O(log E)), 두 번의 Find 함수를 실행하고(O(2log V) = O(log V)), 경우에 따라 Union 함수를 실행한다(O(1)). 결국 최종적인 시간복잡도는 O(E log E) = O(E log V)가 된다.

\*과제 명세에 기재된 대로 그래프 정보가 담긴 파일은 .cpp 파일이 있는 폴더 기준 ./Graphs\_HW4 디렉토리에 저장하여 문제를 해결했습니다.