CSE3018: 알고리즘 설계와 분석

**HW4: Minimum Spanning Tree 알고리즘의 구현**

2023년 6월 24일

**1. 실험 환경**

OS: macOS 13.4.1 (22F82)

CPU: 2.6 GHz 6-Core Intel Core i7

Memory: 16 GB 2400 MHz DDR4

Compiler: Xcode Version 14.3 (14E222b)

**2. Code 구현 내용**

구현 결과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 파일 이름 | 작동여부 | MST weight | 수행시간 | K scanned |
| HW3\_email-Eu-core | YES | 3110161 | 7176 | 25571 |
| HW3\_com-dblp.ungraph | YES | 2747895457 | 411609 | 1049833 |
| HW3\_com-amazon.ungraph | YES | 2729670156 | 435693 | 925857 |
| HW3\_com-youtube.ungraph | YES | 14578691475 | 1.51674e+06 | 2987621 |
| HW3\_wiki-topcats | YES | 5351181035 | 1.72398e+07 | 28416527 |
| HW3\_com-lj.ungraph | YES | 28308045762 | 2.25383e+07 | 34681184 |

시간을 측정할 때에는 명세서에서 지시한 대로 minimum spanning tree 구축에 소요된 시간만 측정하기 위하여 Kruskal MST 함수 내부에 코드를 삽입하였다. Kruskal MST 함수 내에서는 전체 connected edge를 계산하기 위한 반복문이 함께 시행되므로 main 함수에서 측정할 시 추가적인 시간이 소요된다.

추가적으로 주어진 test file들이 직접 결과를 테스트해보기엔 사이즈가 방대하여 간단하게 강의자료에 있는 예제를 이용하여 코드가 잘 작동하는지를 확인해보았다.

A screenshot of a computer

Description automatically generated with medium confidenceA picture containing text, font, screenshot

Description automatically generatedA screenshot of a computer

Description automatically generated with low confidence

순서대로 command.txt, test.txt, test\_result.txt 로, 결과가 잘 나오는 것을 확인할 수 있다.

전체적인 코드의 구성 및 알고리즘은 강의시간에 배웠던 내용과 강의자료에 수록된 함수를 참고하여 그를 기반으로 스스로 코드를 작성할 수 있도록 하였다.

**A screen shot of a computer code

Description automatically generated with low confidence**A screenshot of a computer code

Description automatically generated with low confidence

위와 같이 graph를 구성하는데 필요한 구조체를 선언하였으며 subset의 경우 전체 weight의 sum과 vertex의 수를 추가로 구현하여 결과값 출력을 용이하도록 하였다. Weight\_sum은 공지사항과 같이 total\_weight이므로 int64\_t형으로 저장하였다.

**A screenshot of a computer program

Description automatically generated with medium confidence**

Union 함수 내에서는 subset에 추가로 구현한 변수에 대한 처리가 필요하다. 따라서 매 union 시 weight을 더하는 부분과 vertex수를 합산하는 부분을 추가로 구현하였다. 외에 다른 알고리즘은 강의 시간에 배운 것과 같이 구현하였다. find함수는 recursive하게 parent를 찾아내 반환해주며 union시에는 두 개의 parent 중 rank가 큰 쪽에 다른 쪽을 삽입하므로 큰 쪽의 정보들을, 즉 weight\_sum과 cnt\_vertices를 업데이트한다. Find와 union에서 필요한 시간 복잡도가 각각 O(logV)이다.

A screenshot of a computer program

Description automatically generated with medium confidence

Adjust와 heapsort 역시 강의시간에 배운 바와 같은 알고리즘으로 구현하였다.

A picture containing text, screenshot, font, algebra

Description automatically generated

kruskalMST에서 추가적으로 구현한 부분은 connected에 관한 부분이다. 본 과제에서는 모든 vertex가 하나의 graph로 완성되는지 여부를 알 수 없으며, 총 connected된 수를 결과값으로 출력할 필요가 있다. 따라서 int 배열을 이용해 이를 따로 저장할 수 있는 변수를 추가적으로 구현하였다. Result는 결과 graph를 저장하기 위함인데, 본 과제에서는 결과값에 대한 출력은 필요하지 않다.

A picture containing text, font, screenshot, algebra

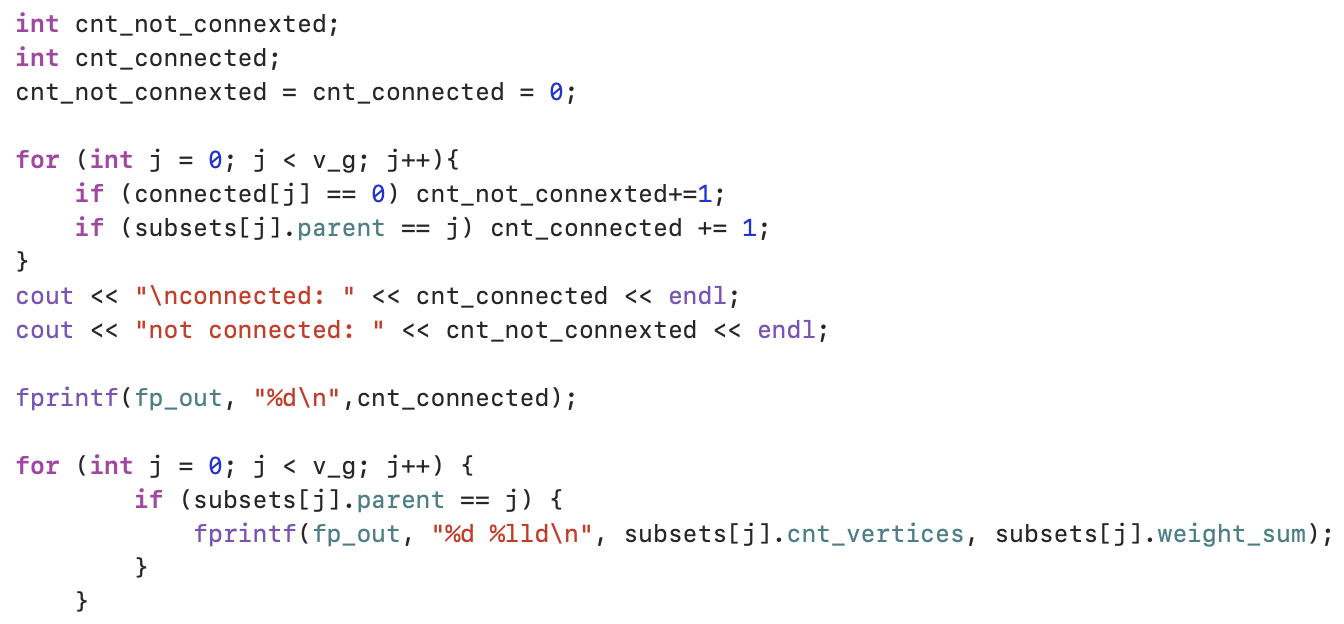
Description automatically generated

앞서 구현한 heapsort를 이용해 그래프를 sort한다. 이때 heap sort에 필요한 시간이 O(ElogE)이다.

A screenshot of a computer program

Description automatically generated with medium confidence

마찬가지로 앞서 구현한 union 을 이용해 MST를 찾는다. while문을 이용해 각 edge의 parent 서로 다른 subset에 있으면 union한다. 여기까지가 MST를 완성하는데 필요한 부분이므로 시간 측정은 여기까지로 한다. K scanned는 while loop가 시행된 수와 같으므로 while loop이 종료된 후의 idx값과 같다.



다음은 connected된 graph의 수를 구하기 위한 부분이다. 앞서 선언한 int 배열 connected와 subset을 살펴본다. Subset의 부모와 자기 자신이 같은 vertex를 connected 된 것으로 판단하고 +1을 해준다.

마지막으로 총 vertex의 개수만큼 foor loop을 돌며 해당 vertex가 root일 경우 총 vertex의 개수, cnt\_vertices와 총 weight, weight\_sum을 output file에 저장해준다.

Kruskal mst 함수에서는 heap sort를 수행하는 시간, k\_scanned번의 while loop을 완료하는데 필요한 시간이 필요하다. While loop 내부에서는 find함수가 시행되며 위에서 설명한 것과 같이find에는 O(logV) 가 걸리므로 총 필요한 시간은 O(E + k\_scanned \* log(v) ) 가 일반적이다. 이때, k\_scanned는 최대 E의 값을 가질 수 있으므로 본 프로그램의 최종 시간 복잡도는 이론적인 시간 복잡도와 같은 O(ElogV)임을 확인할 수 있다. 위에서 서술한 결과값을 계산해보면 이와 같은 결과를 확인할 수 있다.