Final Project of Discrete Mathematics

STEINER TREE

2019/12/19

2019/12/30 UPDATED

Steiner Tree

Given an **undirected graph G(V, E)** with non-negative edge weights and **terminals T**.

The Steiner tree problem in graphs requires **a tree of minimum weight that contains all terminals** (but may include additional vertices).

Problem Statement

The Steiner tree problem in this project can be divided into **Classical part** and **Euclidean part**.

You need to use **C/C++** or **Python** to implement two **approximate algorithms** to solve each part.

In Python, you can use **networkx 2.4** to construct graph easily, but no other function can be called (ex: steiner_tree()).

In C++, you can use Boost to get basename.

Other third-party libraries are NOT allowed.

Project Environment

Ubuntu 18.04 / i5-8400 / 16gb memory

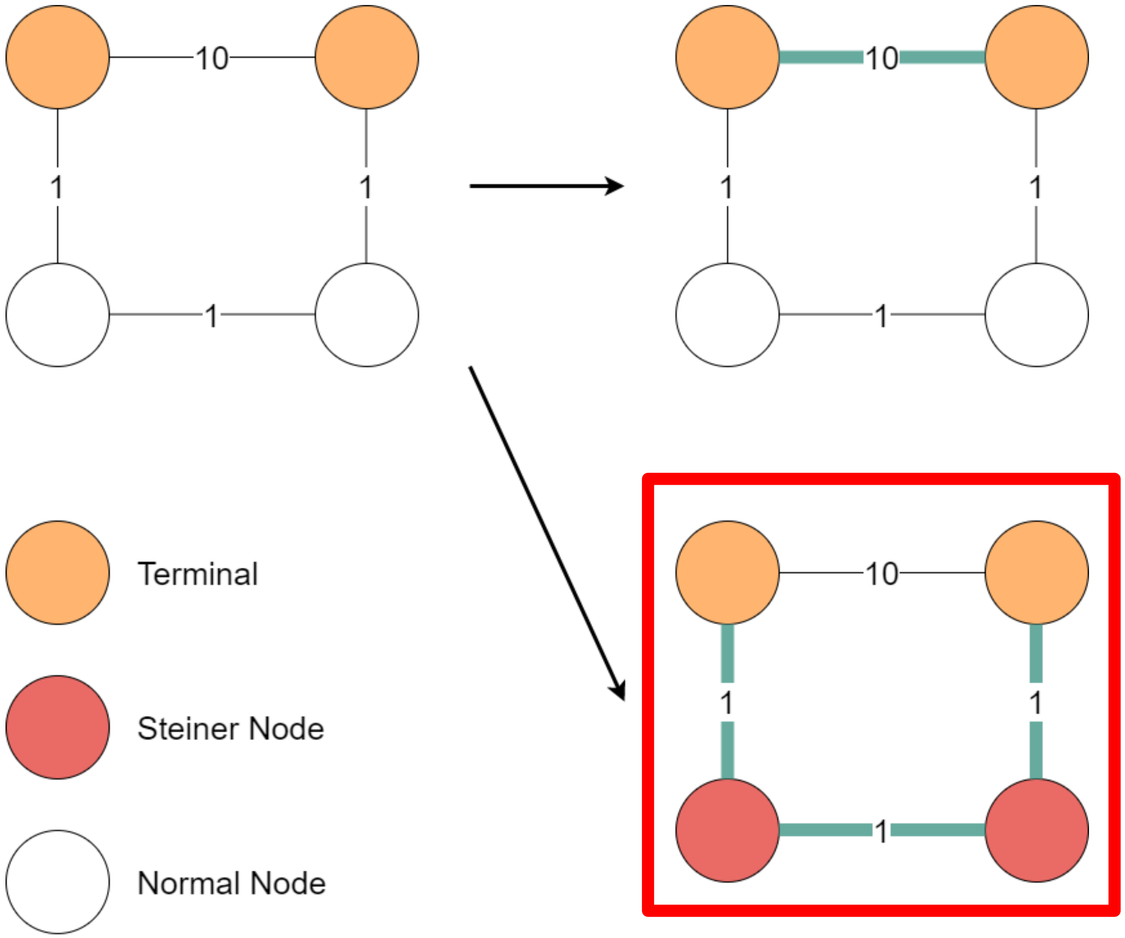
Python:

- Python3.6, networkx2.4

C/C++:

- gcc/g++ 7.4.0

Classical Steiner Tree



minimum spanning tree

Steiner tree

Classical – I/O

The program takes two command arguments, which indicates the path of graph G and terminals T.

C/C++:

```
./classical testcase/classical/b01.stp testcase/classical/b01.stp.terminals
```

Python:

```
python3 classical.py testcase/classical/b01.stp testcase/classical/b01.stp.terminals
```

The program should output one file called b01.stp.outputs ([filename].outputs).

Classical – Input

G

b01.stp

edge 2 8 8.0 weight

2 21 7.0

2 32 2.0

T

b01.stp.terminals

2 terminal

8

Classical – Output

Output
b01.stp.outputs

```
2 8 edge

2 21

2 32

…
```

Output file means the approximate Steiner tree your algorithm found.

output your result to:
output/b01.stp.outputs

Classical – Grading Policy

Baseline (35%, timeout=10s):

Your cost of approximate Steiner tree need to pass the 1.2*(cost from an approximate algorithm in networkx).

Rank (15%): Time * Cost

	Time ↓	Cost ↓:
Top 25% :	4 points	4 points
Top 50% :	3 points	3 points
Top 75% :	2 points	2 points
Others :	1 points	1 points

Time * Cost ↑:	
Top 25% :	15%
Top 50% :	10%
Top 75% :	5%
Others :	0%

Euclidean Steiner Tree

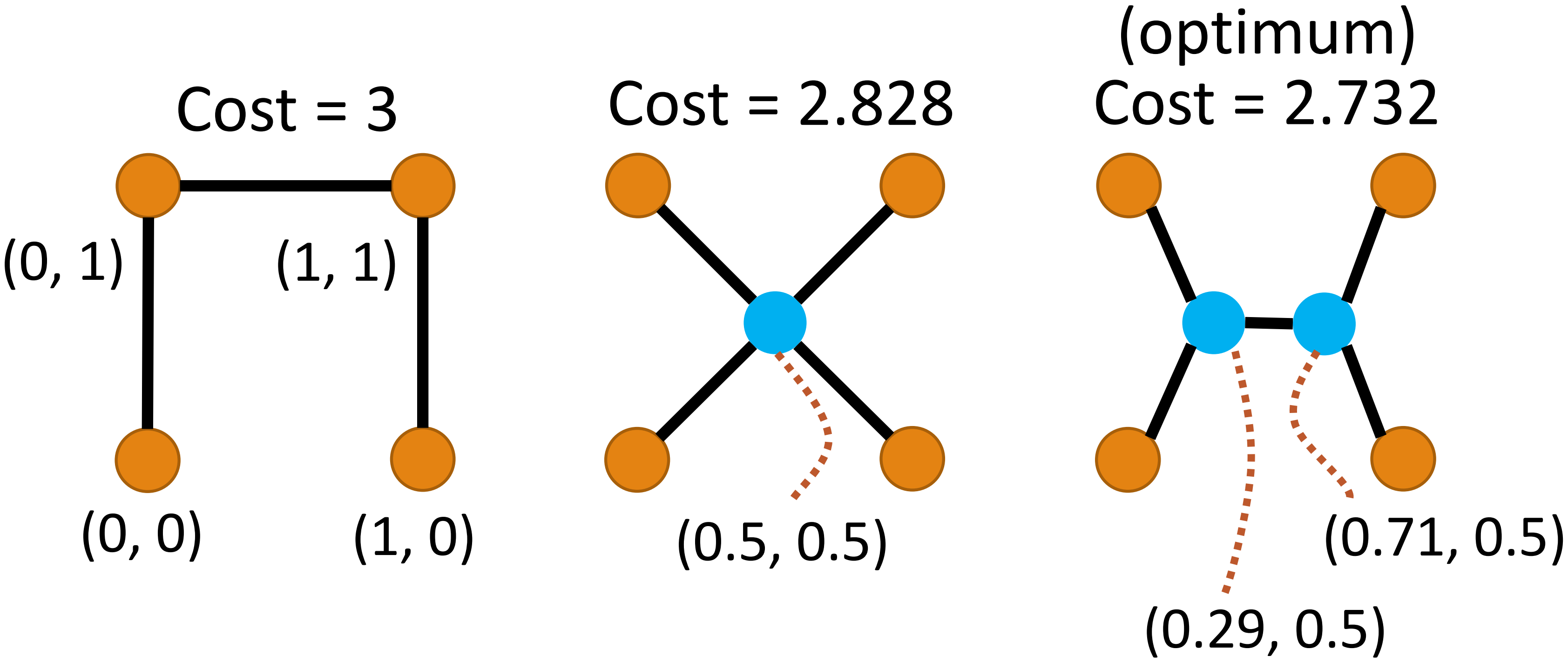
Given **N nodes (terminals)** in the **3-dimensional space**, the goal is to find the minimum spanning tree using all N nodes as well as extra Steiner nodes from the 3-dimensional space.

Also known as Geometric Steiner Tree problem.

Ref: https://en.wikipedia.org/wiki/Steiner_tree_problem
http://www.maths.dur.ac.uk/Ug/projects/highlights/CM3/Soothill_Steiner_talk.pdf

Euclidean Steiner Tree

Consider a 2-D case.



Cost = 3

(0, 1) (1, 1)

(0, 0) (1, 0)

Cost = 2.828

(0.5, 0.5)

(optimum)
Cost = 2.732

(0.71, 0.5)

(0.29, 0.5)

● Terminals ● Steiner nodes

Euclidean – I/O

The program takes one command argument, which indicates terminals.

C/C++:

```
./euclidean testcase/euclidean/1.stp
```

Python:

```
python3 euclidean.py testcase/euclidean/1.stp
```

The program should output one file called 1.stp.outputs ([filename].outputs).

Euclidean – Input

Terminals

(3-dimensional data)

1.stp

4 (number of terminals)

0.000 0.000 0.000

1.000 0.000 0.000

1.000 0.707 0.707

0.000 0.707 0.707

Euclidean – Output

Output

1.stp.outputs

0.711 0.353 0.353 (Steiner node)

0.289 0.353 0.353

1-6, 2-5, 3-5, 4-6, 5-6 (edges)

Output file means the approximate Steiner tree your algorithm found.

Please label the N terminals from 1 to N by the input order, and Steiner nodes should be labeled after the terminals.

output your result to:
output/1.stp.outputs

Euclidean – Output (Example)

- Terminals (input)

4	Node Label
0.000 0.000 0.000	1
1.000 0.000 0.000	2
1.000 0.707 0.707	3
0.000 0.707 0.707	4

- Steiner nodes you find (You don’t need to output the terminal nodes)

0.711 0.353 0.353	5
0.289 0.353 0.353	6

- Edges

1-6, 2-5, 3-5, 4-6, 5-6

Euclidean – Output (Example)

(input file)
這一題中你只需要讀取一個檔案，格式如右邊表示。
第一行表示terminals的數量，後面每一行就是一個terminal的三維座標。

4	Node Label
0.000 0.000 0.000	1
1.000 0.000 0.000	2
1.000 0.707 0.707	3
0.000 0.707 0.707	4

(output file)
只需要輸出你找到的Steiner nodes座標就好，不需要輸出terminals。
再根據terminals輸入順序與output出來的Steiner nodes的順序對這些nodes做編號(從1開始編號)，並由這些編號輸出edges。

0.711 0.353 0.353	5
0.289 0.353 0.353	6
1-6, 2-5, 3-5, 4-6, 5-6	

Euclidean – Grading Policy

Baseline (35%, timeout=10s):

Your cost of approximate Steiner tree need to pass the 1.2*(cost from an existing approximate algorithm).

Rank (15%): Time * Cost

	Time ↓	Cost ↓:
Top 25% :	4 points	4 points
Top 50% :	3 points	3 points
Top 75% :	2 points	2 points
Others :	1 points	1 points

Time * Cost ↑:	
Top 25% :	15%
Top 50% :	10%
Top 75% :	5%
Others :	0%

Ref: Warren D. Smith. How to find Steiner minimal trees in euclidean d-space. Algorithmica (1992) Volume 7, Issue 1-6, pp 137-177

Overall Grading Policy

Classical: 5 test cases

Euclidean: 5 test cases

Each test case must be finished in 10 seconds and has complete 50%, so the final score of the project will be:

(all 10 test cases score) / 5 = final score

I/O

```
|- classical.cpp
|- euclidean.py
|- Makefile (if needed)
|- output
        |- b01.stp.outputs
        |- 1.stp.outputs
|- testcase
        |- classical
                |- b01.stp
                |- b01.stp.terminals
        |- euclidean
                |- 1.stp
```

兩題input檔案都各自放在
testcase/classical/
testcase/euclidean/
輸出的結果請放在
output/資料夾下

假設輸入檔案是
testcase/classical/b01.stp
請記得需要做字串處理，先取出尾巴的b01.stp
變成output/b01.stp.outputs
輸出

Submission

A zip file with your student ID that contains only necessary files.

For example (ID: 0750730):

0750730.zip

|- classical.cpp

|- euclidean.py

|- Makefile (if needed)

Timeline

01/06 23:59 | Checkpoint

01/13 23:59 | Deadline

Networkx: Construct Graph

https://networkx.github.io/documentation/stable/tutorial.html

```
import networkx as nx
G = nx.Graph()
G.add_edge(2, 8, weight=8.0)
G.add_edge(2, 21, weight=7.0)
G.add_edge(2, 32, weight=2.0)
```

b01.stp:

```
2 8 8.0
2 21 7.0
2 32 2.0
```

Networkx: Access Graph

```
print(G[8])

>> AtlasView({2: {‘weight’: 8.0}})

print(G[8][2])

>> {‘weight’: 8.0}

print(list(G.nodes()))

>> [2, 8, 21, 32]

print(list(G.edges()))

>> [(2, 8), (2, 21), (2, 32)]
```

b01.stp:

```
2 8 8.0

2 21 7.0

2 32 2.0
```

Makefile

https://mropengate.blogspot.com/2018/01/makefile.html

利用make程式讀取Makefile檔案來自動化建構軟體

Example Makefile for this project:

```
all: classical euclidean
classical: classical.cpp
    g++ classical.cpp –o classical
euclidean: euclidean.cpp
    g++ euclidean.cpp –o euclidean
```

MinGW Distro

https://nuwen.net/mingw.html

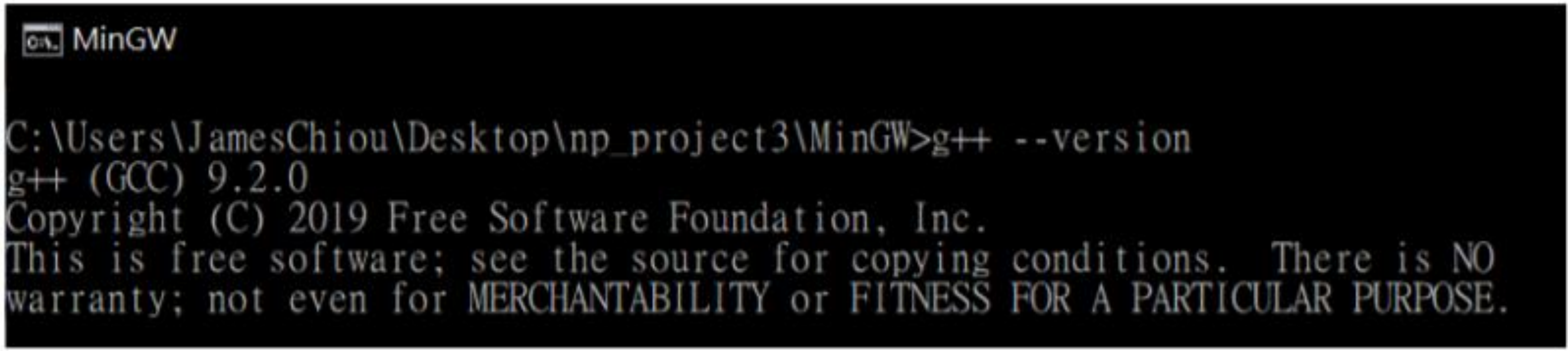
在Windows環境下complie C/C++程式

1. 下載mingw-17.1-without-git.exe (45.1 MB)

2. 解壓縮

3. 在解壓縮目錄下找到open_distro_window.bat

4. 開啟之後打g++ --version，即可確認版本



```
C:\Users\JamesChiou\Desktop\np_project3\MinGW>g++ --version
g++ (GCC) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Boost – 安裝

1. 用Linux (Ubuntu 18.04)環境:
   - 打開terminal後輸入以下指令安裝
   - sudo apt-get install libboost-all-dev

2. 用Windows (MinGW Distro)環境:
   - 自帶不需安裝

Boost – 使用

在Linux, Windows, MacOS下的路徑分隔符號不盡相同，因此可以利用boost中的filesystem來取得檔案名稱。

利用以下指令可簡潔快速的得到路徑中的檔名 (主要用在本次專題輸出output)。

```
#include <boost/filesystem.hpp>
namespace BFS = boost::filesystem;
std::string filename = BFS::path(argv[1]).filename().string();
```

Boost – 編譯

使用boost需要C++並且在compile將其link，可利用以下單行指令達成：

```
g++ euclidean.cpp -o euclidean -lboost_system -lboost_filesystem
```

代表去link安裝好的boost和boost_filesystem。

Evaluation

我們會釋出最後評分使用的Python程式(evaluation.py)

同學可以將evaluation.py這個檔案和你們的程式放在同個資料夾，並直接執行python evaluation.py試試看你們的程式是不是能正確輸入輸出。

這隻程式會自動去編譯與執行兩題的程式，並驗證你輸出的Steiner Tree是否正確，以及計算cost和執行時間。