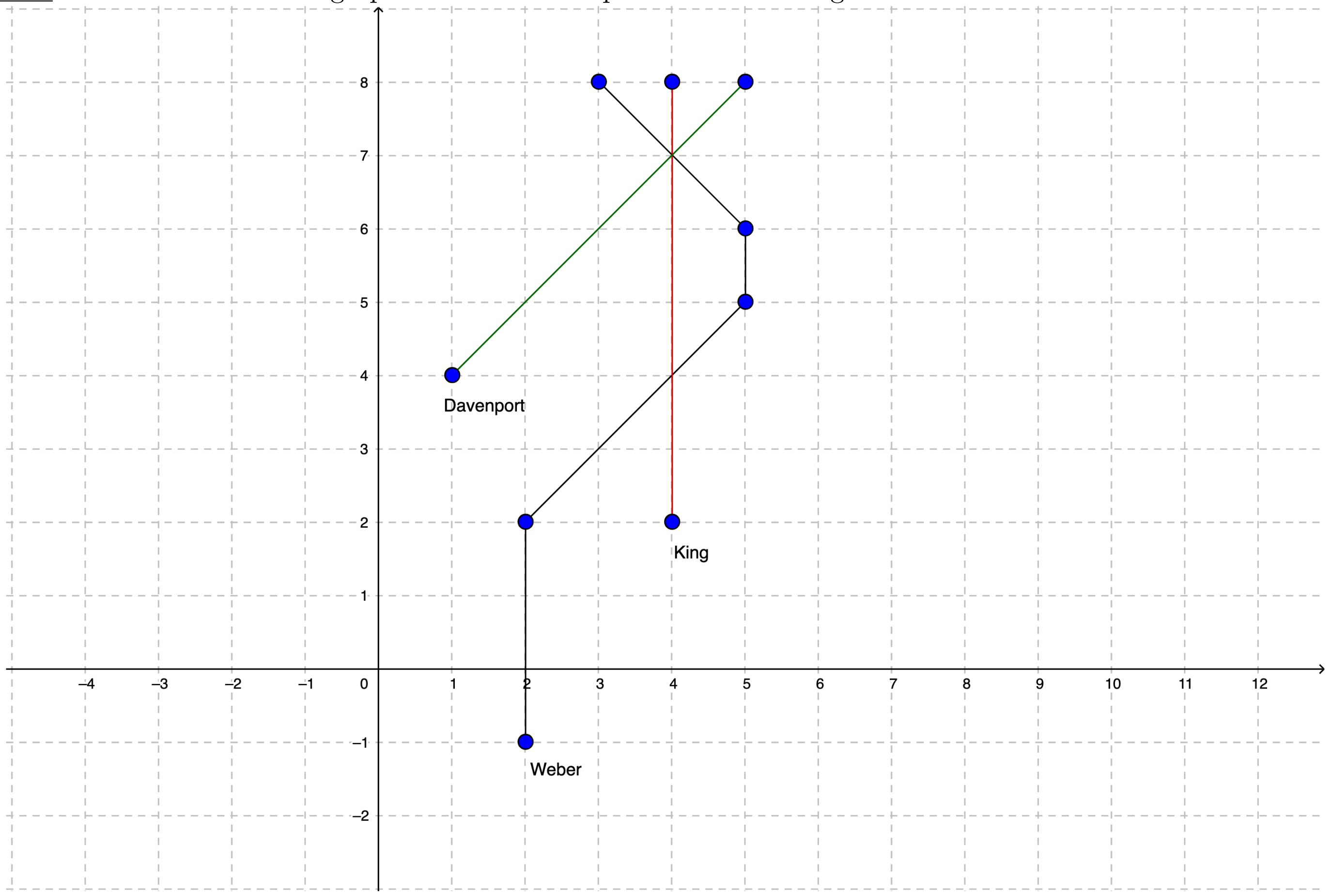
1 Assignment 1: Frequently Asked Questions

We created the following FAQ based on the many questions about A1 edge cases on Piazza (the post IDs are also given below). The goal is to clarify and simplify the specification so that you only have to deal with a minimal set of edge cases.

Question 0. Can we have a graphical representation of a set of streets?

Answer. Please see below the graph for the streets specified in the assignment1 PDF.



Question 1. Are empty street names allowed? What about street names with only white spaces? What about leading and trailing white spaces? @177

Answer. We do NOT allow street names with no characters whatsoever or only with white spaces. Street name must have at least one alphabet character (i.e., [a-zA-Z]). Leading and trailing white space characters are also NOT allowed.

Question 2. What is the definition of a line segment? @169

Answer. A line segment is the line connecting two consecutive coordinates as specified in one of the input commands. For example, the street specified by 'add "Street A" (0,0) (0,1) (0,2) (0,3)' has 3 line segments: Line segment 1: $\{(0,0),(0,1)\}$, Line segment 2: (0,1), (0,2), Line segment 3: (0,2), (0,3)

Question 3. Can we have overlapping line segments or streets? @183, @140

Answer. We disallow overlapping streets or line segments. After considering the many questions on

Piazza regarding overlapping streets, we decided we would not test for overlapping streets or line segments. Your code does not have to explicitly reject input that has overlapping streets or line segments, nor does it have handle it correctly since this will not be tested in the program.

We considered the possibility of requiring your code to explicitly reject overlapping line segments, but quickly realized that this will be quite a bit of additional work for you. Hence, we rejected this idea of requiring your code to explicitly reject overlapping line segments. Since we don't test for it, you don't have to worry about it.

Question 4. Do we allow disconnected graphs? @170, @181

Answer. Yes, this is allowed. We expect your code to handle them exactly as you would do with connected graphs (per the specification in the original assignment description).

Question 5. What is an acceptable coordinate input? @179, @172

Answer. Input format should be (integer1,integer2). We don't allow any white space between a matching pair of "(" and ")". Negative numbers should have "-" sign in front of it, positive numbers should not have "+". Examples of acceptable and non-acceptable inputs are shown below.

```
(1,2) - accept
(+1,-1) - not correct. Plus sign in front of 1 is not correct
(1,-       2) - not correct. White space between minus sign and digit is not correct
(1, 2     4) - not correct. Space between ‘2’ and ‘4’ is not correct.
```

Question 6. When does the program terminate?

Answer. The program terminates when it encounters a EOF character, Ctrl+D.

Question 7. What code in your program handles termination? @175

Answer. One option is the following: Sys.exit(0) as provided in the assignment solution skeleton.

Question 8. Should adding a street name that already exists cause an error? @126

Answer. Yes, this should cause an error. You can "mod" an existing street but not "add" it.

Question 9. **(Set ordering)** How should vertices and edges be ordered? The assignment says "Your output has to perfectly match what is expected."

Answer. Since the vertices V and edges E are sets, there is no specification on ordering. You may output the entries in any order. The graph is not directed, so edges themselves, i.e., pairs of coordinates, can be output in arbitrary order as well.

Question 10. **(Intersections)** Should we calculate the intersection points in floating point format or integer?

Answer. The intersections are not necessarily integers. You can, for example, assume that coordinate values within, say, 0.0001 of one another are the same. We will not test your code with weird intersections that are very close to one another.

Question 11. **(White-space)** For the following examples, can you tell me if the code should accept it or reject it?

```
add "weber" 1,2,3,4,5,6 - no brackets
add "weber" (1,2 (3,4) (5,6) - missing bracket
add "weber" (1,2)(3,4) (5,6) - no space between the brackets
add"weber" (1,2) (3,4) (5,6) - missing space between command
add "weber"(1,2) (3,4) (5,6) - missing space after street name
```

Answer. The first and second examples are erroneous and should be rejected. The third example is erroneous because there is no space bewteen the brackets. The fourth and fifth examples are erroneous and should be rejected because there is missing space that does not separate the street from the rest of the command line. This is typical with software – we often ignore the exact number of white-space characters as long as there is at least one white-space between the command and arguments (as well as among the arguments). The same is true for the input to your program as well as the output graph that your program must generate.

Question 12. **(Float format)** How should float values be printed. Is the following a good output?

```
g
V = {
  1: (2.000000,2.000000)
  2: (4.000000,4.000000)
}
E = {
  <1,2>
}
```

Answer. No. You should print all coordinate values rounded to two decimal places. You can do so using `format` function when converting a `float` to a `string`. For example,

```
>>> "{0:.2f}".format(2.0)
'2.00'
>>> "{0:.2f}".format(2.324)
'2.32'
```

Note that it is not necessary to always have two decimal digits. For example, an integer coordinate 2 can be printed as either one of `2`, `2.0`, or `2.00`.

Question 13. Do vertex ids have to be numeric. Should they always be printed in ascending (smallest-to-largest) order?

Answer. No, vertex ids can be arbitrary strings subject to the specification in the assignment.

Question 14. Can a street intersect itself?

Answer. We assumes that streets do not intersect themselves. ~~Note that street segments of different streets may overlap. In the case of overlapping street segments, the end points of the overlapping region are vertices.~~ Please see the new FAQ regarding overlapping streets or line segments.

Question 15. Are street names case sensitive? Do spaces matter in street names?

Answer. Street names are case insensitive. For example, "King street" and "KING street" mean the same street. ~~The street name is composed of all the characters between quotes. If the characters are~~

~~different, the street names are different. This includes whitespace characters as well. For example, "King" is a different from " King".~~ Refer to the new FAQ above for details about white spaces in street names.

Question 16. Are command names lower case?

Answer. Command names are lower case as in all examples in the handout.

Question 17. What characters are allowed in street names?

Answer. Alphabetical and space characters only. No numbers, commas, special characters, unicode characters, etc.

Question 18. How do we find the intersection between two line segments?

Answer. There is an example in Python in the repo `https://git.uwaterloo.ca/ece650-f2021/python-examples/-/tree/master/py`. While this produces an intersection point, be careful and consider if the intersection point makes sense or is valid.

Question 19. What does the format of the edges mean?

Answer. The edges are represented as a connection or edge between two verticies. In the following example the edge is a connection between vertex 1 and 2.

```
g
V = {
  1: (2.000000,2.000000)
  2: (4.000000,4.000000)
}
E = {
  <1,2>
}
```

2 Assignment 1: Sample Test Cases

Below find some sample test cases to better test your code against edge cases. Note that we have simplified the specification of the assignment quite a bit and so it should be easier to handle these cases. Please also see the new FAQ.

2.1 add

```
add "weber" 1,2,3,4,5,6 - no brackets - error

add "weber" (1,2 (3,4) (5,6) - missing bracket - error

add "weber" (1,2)(3,4) (5,6) - no space between the brackets - error

add"weber" (1,2) (3,4) (5,6) - missing space between command - error

add "weber"(1,2) (3,4) (5,6) - missing space after street name - error

add " " (1,2) (3,4) (5,6) - empty Street name - error

ADD "weber" (1,2) (3,4) (5,6) - invalid command - error
```

```
add "weber" (-1,+2) (3,4) (5,6) - invalid coordinates
        - we don't allow +sign in coordinates, we allow '-' sign in front of numbers - error

add "weber" (  -   1,2) (3,4) (5,6) - invalid coordinates
        - we don't allow white space between the '-' sign and the number - error

add "weber" (1,2) - invalid command - error - there should be at least one pair of coordinates

add "weber*" (-1,2) (3,4) (5,6) - error - invalid street name, contains special characters

add "St. Jacob's Market Street" (-1,2) (3,4) (5,6) - error - invalid streetname, contains special charac

add "2nd Avenue" (-1,2) (3,4) (5,6) - error - invalid street name, contains numbers

add "weber" (-1,2) (3,4) (5,6) - valid command

add "king" ( -1 , 2) ( 3 , 4) ( 5 , 6 ) - invalid command
            - There should not be a space in between a pair of matching "(" and ")"

add "weber" (1,2) (3,4) (5,6) - adding "weber" street again should throw error

add " weber" (1,2) (3,4) (5,6) - there is a leading white space - should throw error - same street

add " Weber" (1,2) (3,4) (5,6) - there is a leading white space and street name is case insensitive
        - should throw error - same street
```

2.2 mod

```
MOD "weber" (1,2) (3,4) (5,6) - invalid command - error - the commands are case sensitive

mod "weber"  (1,2) (3,4) (5,6) - valid command

mod "Weber" (1,2) (3,5) (7,9) - valid command
           the street name is case insensitive, so street exists and you can modify the coordinates

mod "univ avenue" (1,2) (3,4) (5,6)
    - valid command but should throw an error if the street does not exist
```

2.3 rm

```
rm "weber" (1,2) (3,5) (7,9)- invalid command, cannot have 2 arguments

rm "weber" - valid command

rm "test street" - valid command but should throw an error if "test street" does not exist
```

2.4 gg

```
gg "weber"- error - not a valid command
gg - valid - should print edges and vertices
```

3 Examples Inputs and Corresponding Outputs

3.1 Example 1

```
gg
#If the graph is empty, print empty graph

V = {
}
E = {
}
```

3.2 Example 2

```
add "Weber Street" (2,-1) (2,2) (5,5) (5,6) (3,8)
add "King Street S" (4,2) (4,8)
add "Davenport Road" (1,4) (5,8)
mod "Weber Street" (2,1) (2,2)
gg

V= {
2: (4,2)
5: (1,4)
6: (4,7)
8: (5,8)
10: (4,8)
}
E = {
<2,6>,
<6,5>,
<6,8>,
<6,10>
}

rm "King Street S"
gg

V = {
}
E = {
}
```

3.3 Example 3

```
#Two disconnected graphs

add "test one" (1,1) (4,4)
add "test two" (0,4) (4,0)
add "test three" (9,1) (12,4)
add "test four" (8,4) (12,0)

gg

V = {
  1: (1,1)
  2: (4,4)
  3: (0,4)
  4: (4,0)
  5: (2,2)
  6: (9,1)
  7: (12,4)
  8: (8,4)
  9: (12,0)
  10: (10,2)
}
E = {
  <1,5>,
  <5,2>,
  <3,5>,
  <5,4>,
  <6,10>,
  <10,7>,
  <8,10>,
  <10,9>
}
```

4 Precise Input Specifications

4.1 add/mod

add and ***mod*** take input two types of arguments: 1. ***"streetstring"*** 2. ***list of coordinates***, the commands should be in lower cases.

1. ***streetstring*** is a non-empty string containing only upper and lower case letters along with spaces. A space can only appear in between a pair of consecutive letters, and there can be no consecutive spaces.
2. ***list of coordinates*** contains *coordinates* separated by exactly one space. Where each coordinate is of the form (i_1, i_2), where i_k is an integer.
   - There is no space in between a pair of matching open and close parentheses.
   - The only sign we use is "-" when some i_k is negative.
3. There is exactly one space between ***add/mod*** and ***"streetstring"***, and exactly one space between ***"streetstring"*** and ***list of coordinates***.

4.2 rm

rm take input one type of argument: ***"streetstring"***, the command should be in lower cases.

1. ***streetstring*** is a non-empty string containing only upper and lower case letters along with spaces. A space can only appear in between a pair of consecutive letters, and there can be no consecutive spaces.

2. There is exactly one space between ***rm***/***add*** and ***"streetstring"***.

4.3 gg

gg takes input zero argument, and the command should be in lower cases.