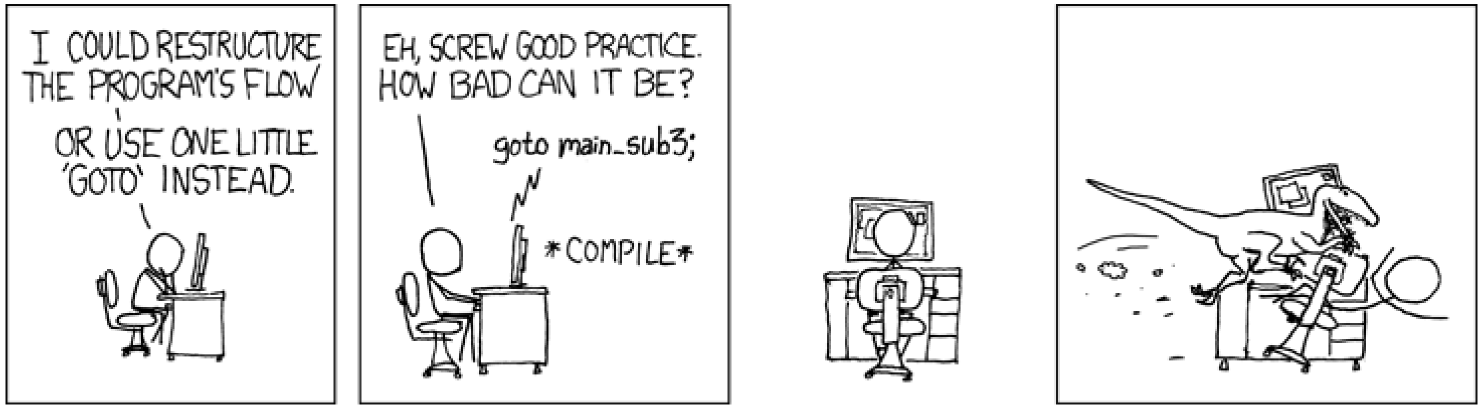
CSc 110, Spring 2017

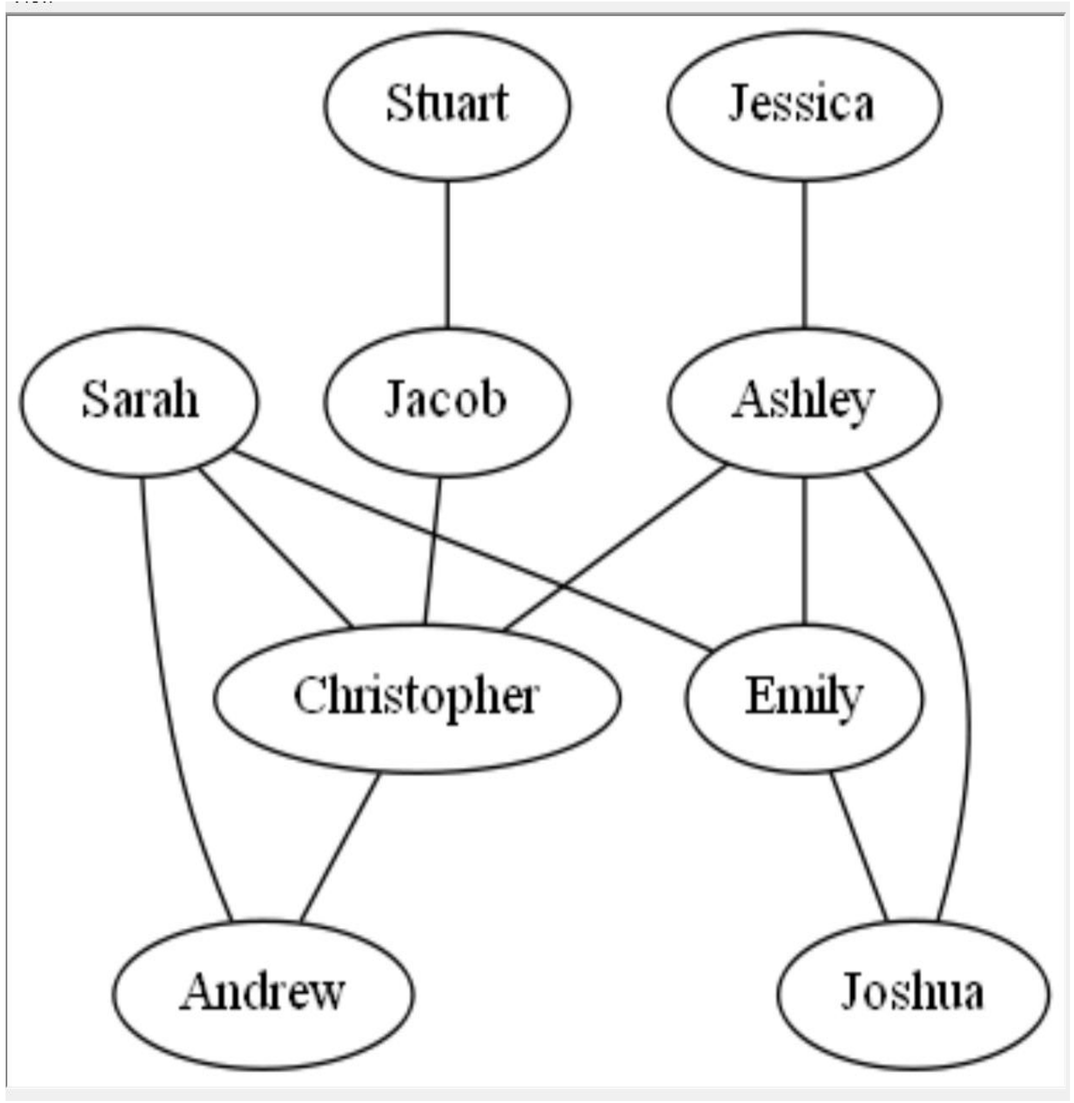
Lecture 31: 2D Structures

Adapted from slides by Marty Stepp and Stuart Reges


I COULD RESTRUCTURE THE PROGRAM'S FLOW
OR USE ONE LITTLE 'GOTO' INSTEAD.
EH, SCREW GOOD PRACTICE. HOW BAD CAN IT BE?
goto main_sub3;
COMPILE

Exercise

- Write a program that allows a user to ask the distance between two people in a network of friends.
  - If person 1 and person 2 are friends then they are at distance 1
  - If person 2 is friends with a friend of person 2 they are at distance 2



```
graph {
  Ashley -- Christopher
  Ashley -- Emily
  Ashley -- Joshua
  Christopher -- Andrew
  Emily -- Joshua
  Jacob -- Christopher
  Jessica -- Ashley
  Sarah -- Andrew
  Sarah -- Christopher
  Sarah -- Emily
  Stuart -- Jacob
}
```

Name 2 friends at distance 1.
Which two people are at the greatest distance?

<u>Ashley</u>	<u>Christopher</u>	<u>Emily</u>
Christopher	Ashley	Ashley
Emily	Jacob	Joshua
Joshua	Andrew	Sarah
Jessica	Sarah	

Note: not all sets of friends shown.

```
graph {
  Ashley -- Christopher
  Ashley -- Emily
  Ashley -- Joshua
  Christopher -- Andrew
  Emily -- Joshua
  Jacob -- Christopher
  Jessica -- Ashley
  Sarah -- Andrew
  Sarah -- Christopher
  Sarah -- Emily
  Stuart -- Jacob
}
```

```
# Reads in a dot file with friendship data
# Version 0: Asks if two people are friends

def main():
    file = open("friends.dot")
    lines = file.readlines()
    friends = create_dict(lines)

    for name in friends:
        print(name, " : ", friends[name])
    name1 = input("Enter a name: ")
    name2 = input("Enter a name: ")

    #Are name1 and name2 friends?
```

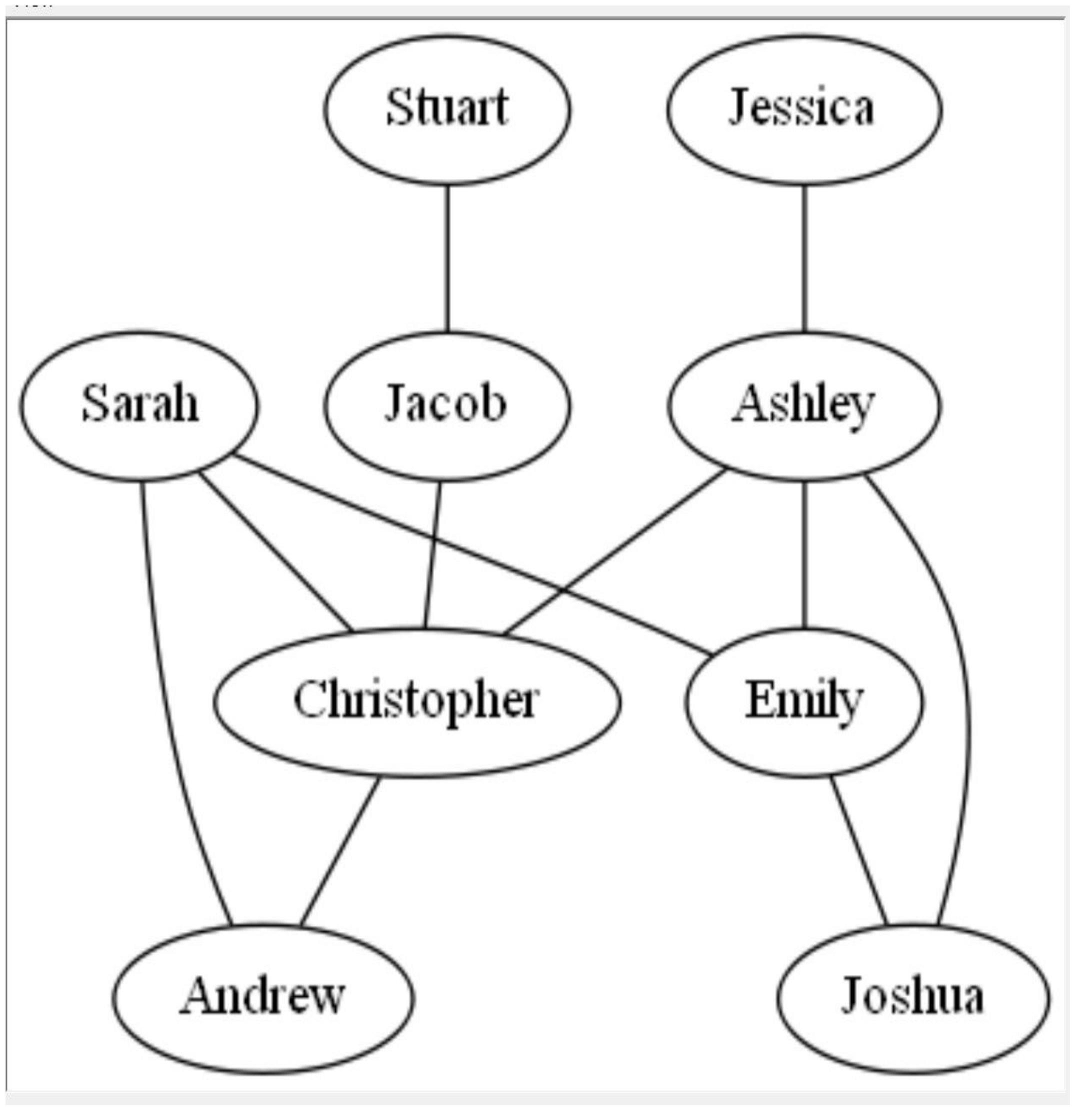
```
# creates and returns a dictionary mapping each person to a
# set of their friends. Creates an entry for name1 to name2
# and name2 to name1.
def create_dict(lines):
    friends = {}
    # skip the first and lst lines as they have dot syntax
    for i in range(1, len(lines) - 1):
        line = lines[i].split()
        name1 = line[0]
        name2 = line[2]
        if (name1 not in friends):
            friends[name1] = set()
        friends[name1].add(name2)
        if (name2 not in friends):
            friends[name2] = set()
        friends[name2].add(name1)

    return friends
```

`friends` dictionary

- The content of the `friends` dictionary is:

```
{
    Stuart  :  {'Jacob'}
    Jacob   :  {'Stuart', 'Christopher'}
    Ashley  :  {'Christopher', 'Emily', 'Joshua', 'Jessica'}
    Sarah   :  {'Christopher', 'Andrew', 'Emily'}
    Jessica :  {'Ashley'}
    Andrew  :  {'Christopher', 'Sarah'}
    Emily   :  {'Ashley', 'Joshua', 'Sarah'}
    Joshua  :  {'Ashley', 'Emily'}
    Christopher  :  {'Jacob', 'Ashley', 'Andrew', 'Sarah'}
}
```


Stuart
Jessica
Sarah
Jacob
Ashley
Christopher
Emily
Andrew
Joshua

Pseudocode for finding the distance – Version1

```
initialize a current set of friends to name1
initialize distance to zero
while name2 not found in current set of friends
    increment the distance
    make a new set of friends from the current set using the dictionary
        to reference the sets of friends
    set the current set of friends to the union of the current set and new set of friends

print the distance
```

Sarah to Joshua

- This works but what if we looked for someone out of the friend network?
- What is the problem with `current_friends`?

```
new_friends
{'Christopher', 'Andrew', 'Emily'}
current_friends
{'Christopher', 'Sarah', 'Andrew', 'Emily'}
new_friends
{'Sarah', 'Ashley', 'Andrew', 'Emily', 'Jacob', 'Joshua',
 'Christopher'}
current_friends
{'Ashley', 'Jacob', 'Joshua', 'Sarah', 'Andrew', 'Emily',
 'Christopher'}
distance is:  2
```

We are never removing names that we have already seen.

Pseudocode for finding the distance – Version2

```
initialize a current set of friends to name1
Initialize a set of already seen friends to name1
initialize distance to zero
while name2 not found in current set of friends and length of current friends not zero
    increment the distance
    make a new set of friends from the current set using the dictionary
        to reference the sets of friends
   already seen friends is assigned to the union of itself and current friends
    set the current set of friends to the new set of friends minus the already seen friends

 if  the length of the current set of friends is not zero
    print the distance
else
   print not connected
```

```
# Reads in a dot file with friendship data - Version2
def main():
    file = open("friends.dot")
    lines = file.readlines()
    friends = create_dict(lines)
      name1 = input("Enter a name: ")
    name2 = input("Enter a name: ")

    #Are name1 and name2 friends?
    current_friends = {name1}
    already_seen = {name1}
    distance = 0
    # stops when the friend is found or there is no possibility of a connection
    while(name2 not in current_friends and len(current_friends) != 0):
        distance += 1
        new_friends = set()
        # builds up a set of the friends of the current friends
        for friend in current_friends:
            new_friends = new_friends | friends[friend]
        already_seen = already_seen | current_friends
        # replaces current friends and gets rid of friends looked at before
        current_friends = new_friends - already_seen

    if(len(current_friends) != 0):
        print("found at distance " + str(distance))
    else:
        print("sorry they are not connected")
```