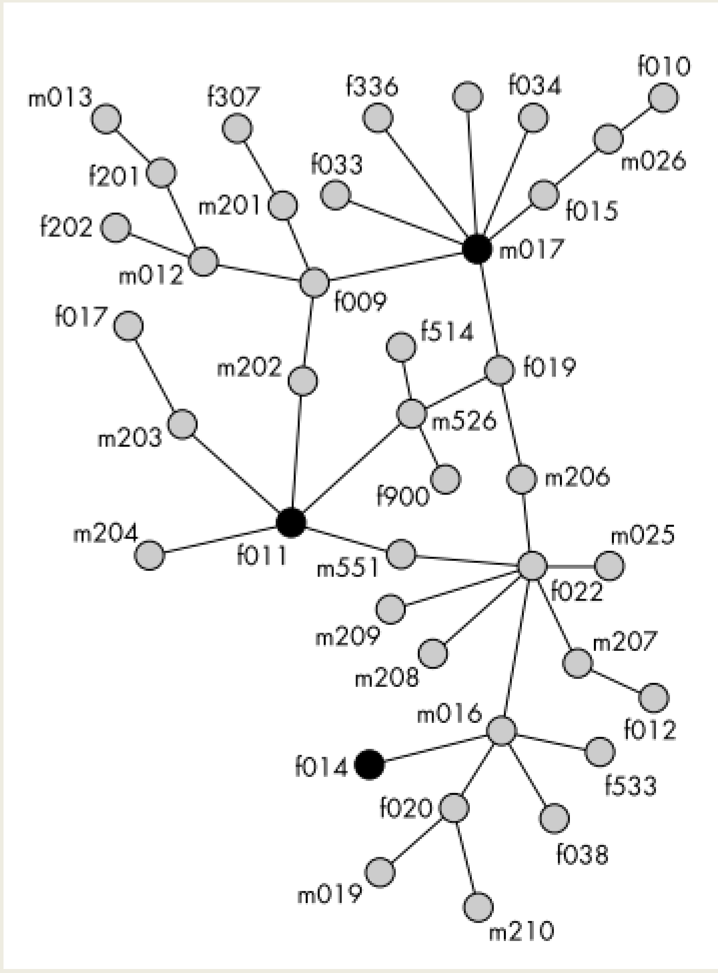
CSCI 2270
Data Structures and Algorithms
graph lecture 1



Elizabeth White
elizabeth.white@colorado.edu
Office hours: ECCS 128
Wed 1-2pm
Fri 2-3pm

Administrivia

Colloquium writeup was due yesterday

Today: STL vectors and STL maps

Last homework on graphs

- Begin today in class with STL classes
- Lab this week: reading the data in from a file to a graph
- HW3: Find shortest-hop path between 2 cities
  - depth first or breadth first search

Friday lecture and lab next week, hacking lab

Next week lecture demo: hash tables and timings

Vectors act like arrays

```
#include <vector>
using namespace std;
int main() {
        vector<int> vobo;
        int k = 0;
        while (k < 10) {
                vobo.push_back(rand() % 100); ++k;
        }
        k = 0;
        while (k < 10) {
                cout << vobo[k] << endl; ++k;
        }
}
```

Vectors and iterators

```
#include <vector>
using namespace std;
int main() {
        vector<int> vobo;
        int k = 0;
        while (k < 10) {
                vobo.push_back(rand() % 100); ++k;
        }
        vector<int>::iterator vobo_it = vobo.begin();
        while (vobo_it != vobo.end()) {
                cout << *vobo_it << endl; ++vobo_it;
        }
}               // iterator acts like a pointer (derefenced with *)
```

Maps store pairs

```
#include <map>
using namespace std;
int main() {
        map<string, int> ages; string name; int age;
        int k = 0;
        while (k < 5) {
                cout << “Enter a person “;
                cin >> name;
                cout << “Enter an age for “ << name << “ “;
                cin >> age;
                ages[name] = age;
        }
```

Maps store pairs

```
#include <map>
using namespace std;
int main() {
          …
          while (ages_it != ages.end()) {
                    cout << ages->first << “ is “ << ages->second <<
                              “ years old “ << endl;
                    ++ages_it;
          }
}         *ages_it is a pair of data: one string and one int
          dereference this iterator with ->
```

Maps

Bursar’s office

maps your IndentiKey to your billing records

Government offices, like IRS

maps your social security number to your tax payments

Phone contact list

maps your friends’ names to their phone numbers

Useful when you want to remember a *relationship between 2 things*

Store city information in vertex.h

```
class vertex    {
        public:
                // …
        private:
                string name_of_city;
                int latitude_degrees;
                int latitude_minutes;
                int longitude_degrees;
                int longitude_minutes;
```

For example:

Cape Town, South Africa: 33 55 S 18 22 E

Caracas, Venezuela: 10 28 N 67 2 W

Graph is vector of vertices plus map of vertices to neighbors

```
class graph {
   public:          …
          graph();
          ~graph();
          void add_vertex(const vertex& v);
          void add_edge(vertex* v, vertex* u, double dist);
   private:
          // list of cities
          vector<vertex> vertices;
          // neighbors of each city
          map<vertex *, vector<vertex *> > edges;
}
```