CityFinder

...

Original from : Mike Precup (mprecup@cs.stanford.edu)
ENJMIN Edition 2016

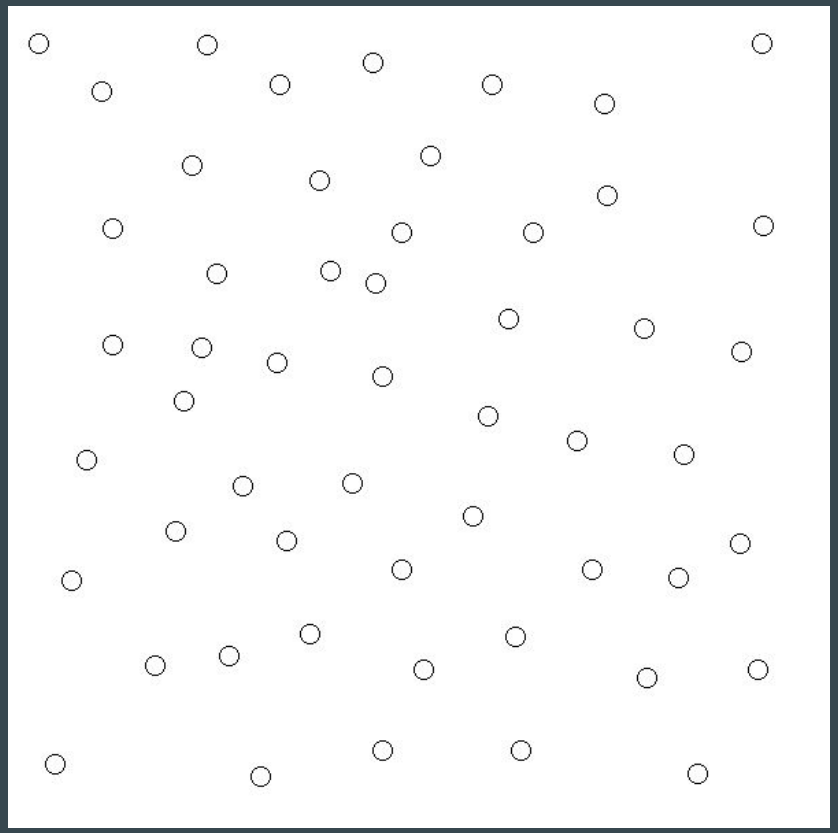
CityFinder

CityFinder answers the question:

Which cities are within `d` miles of city `y`?

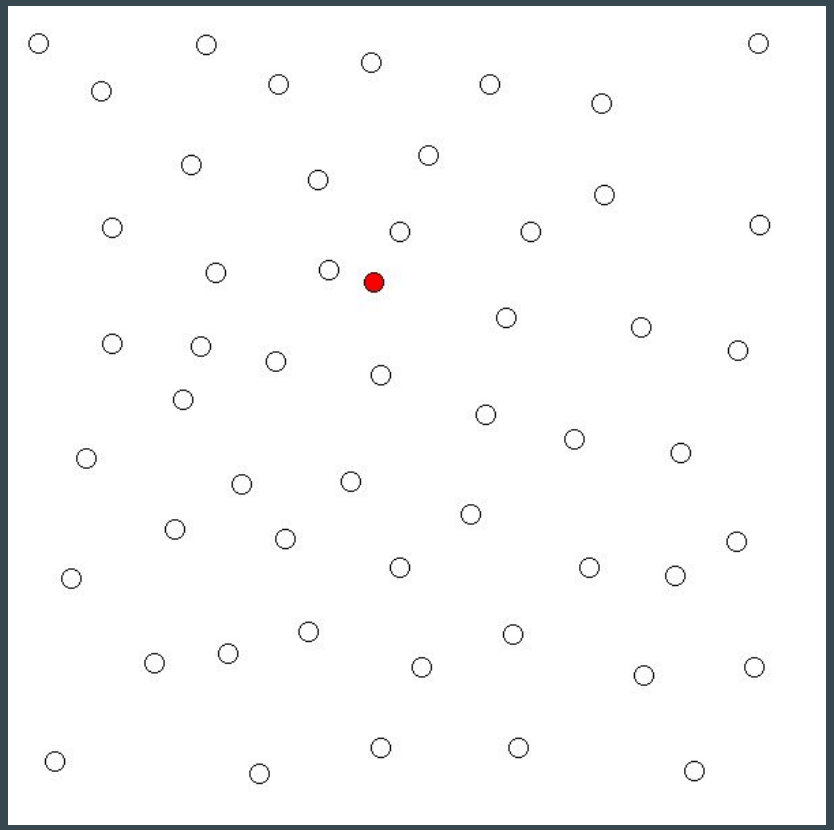
CityFinder

Say that we had a bunch of cities arranged like this:



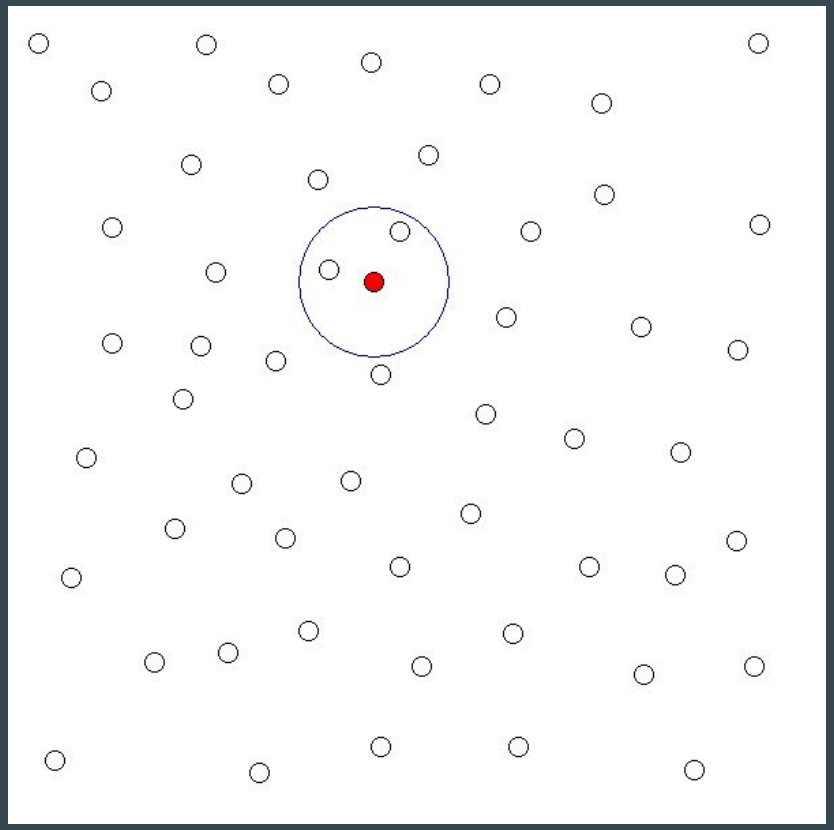
CityFinder

Say that we're considering this city



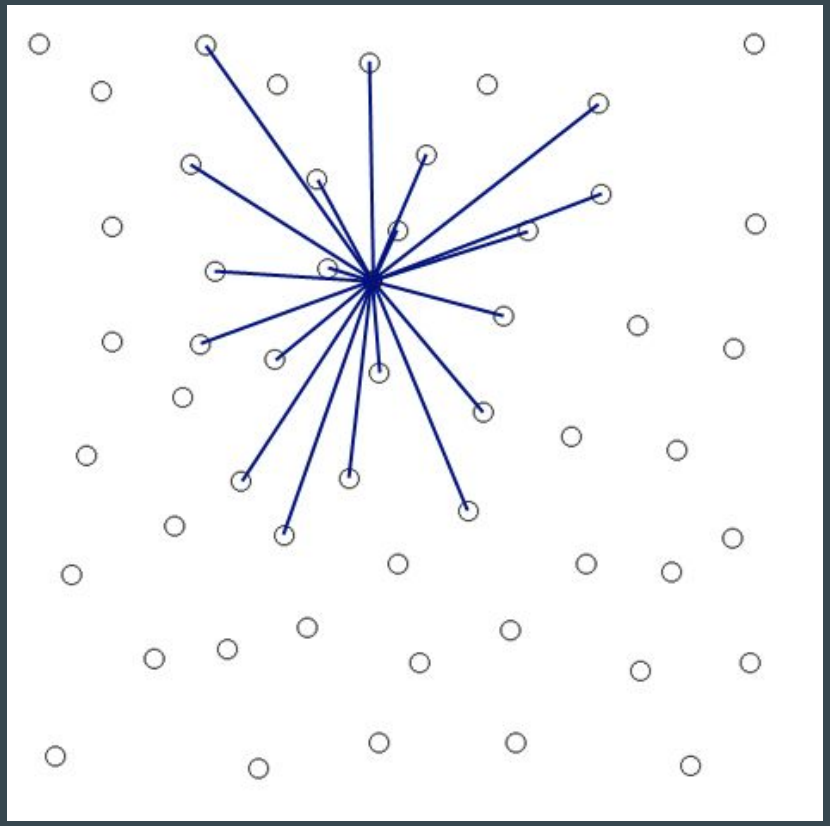
CityFinder

Say we wanted to find the names of all cities within a certain distance of our city



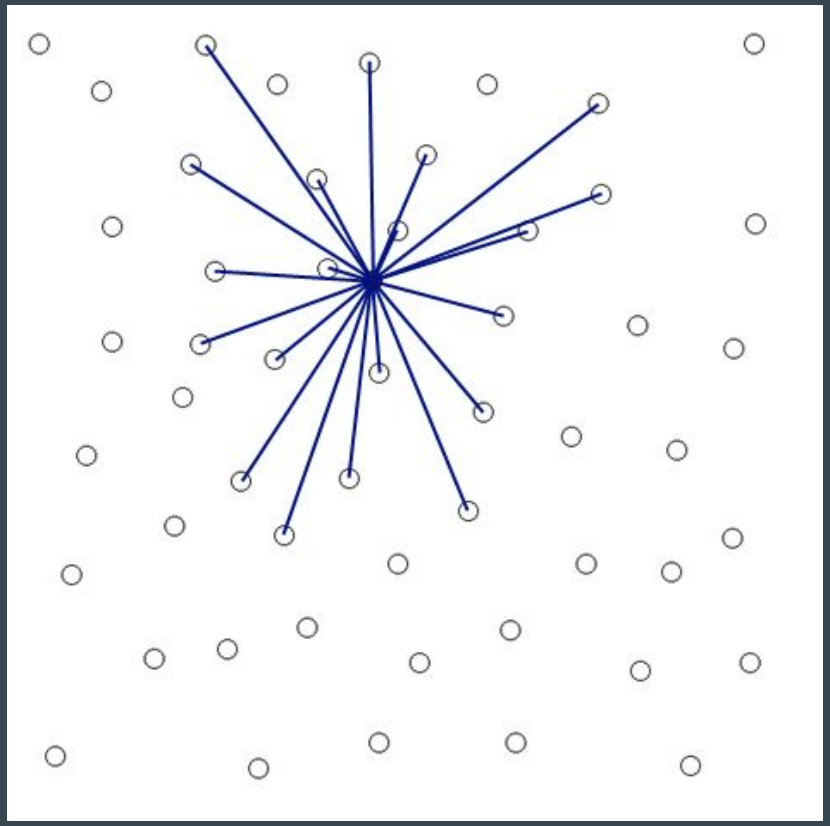
CityFinder

We could compare the distances between our city and every other city



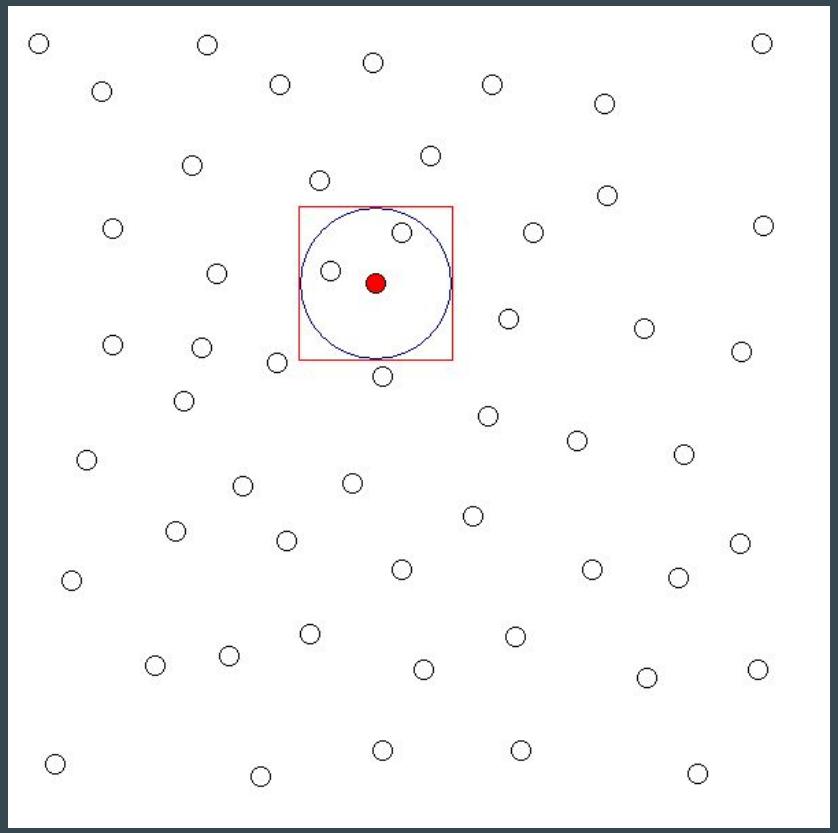
CityFinder

This would be too slow -- computing distance can be expensive!



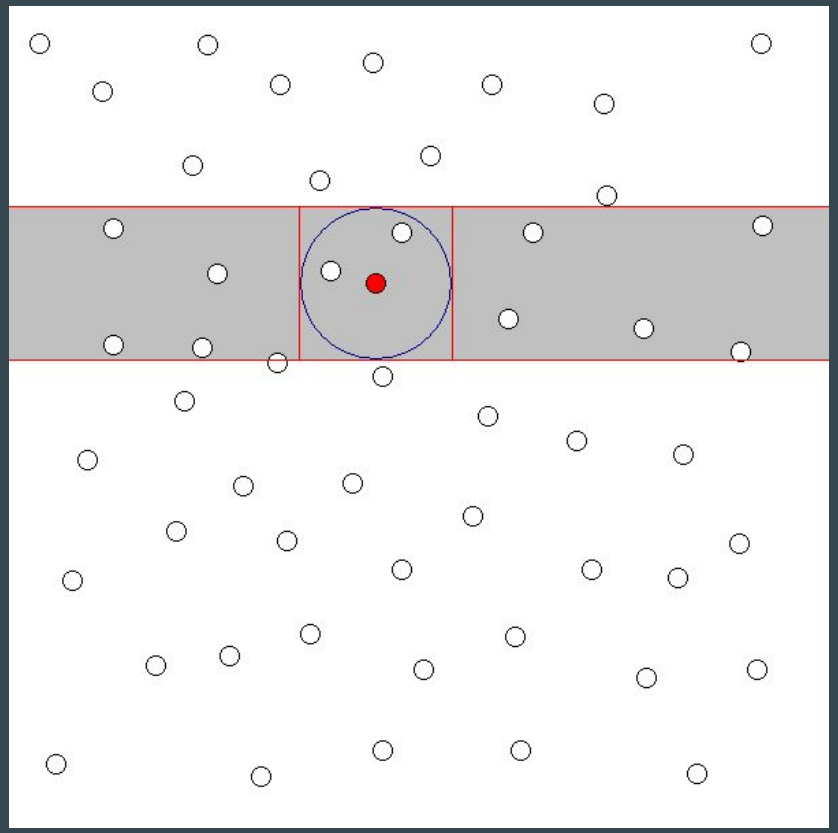
CityFinder

What if we used the same process, but only needed to check a smaller number of cities?



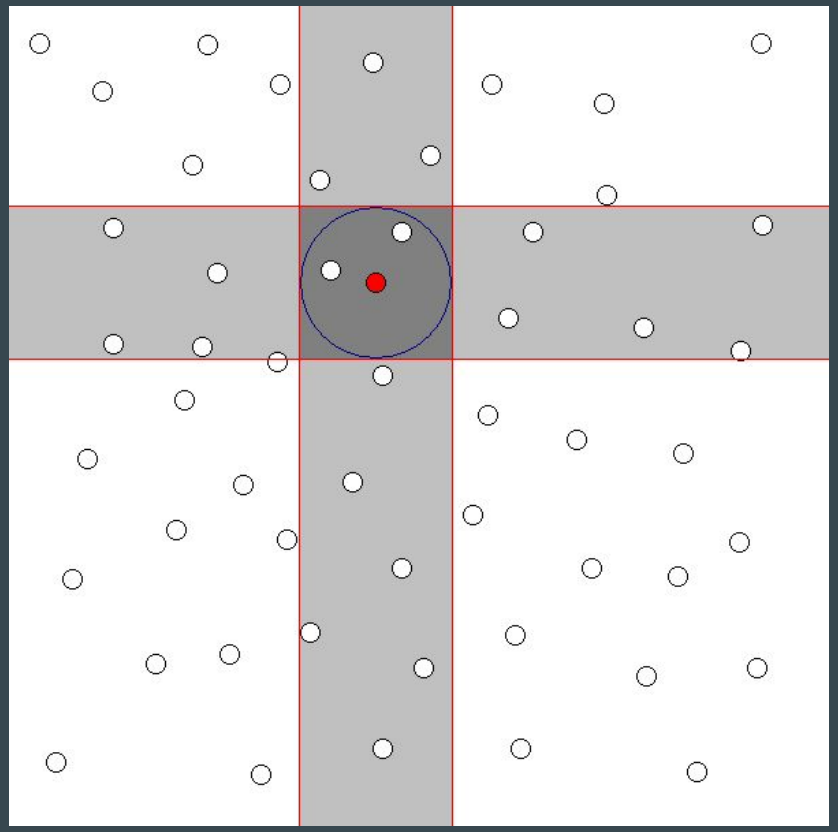
CityFinder

We could start by excluding cities who have y coordinates which are too large or too small



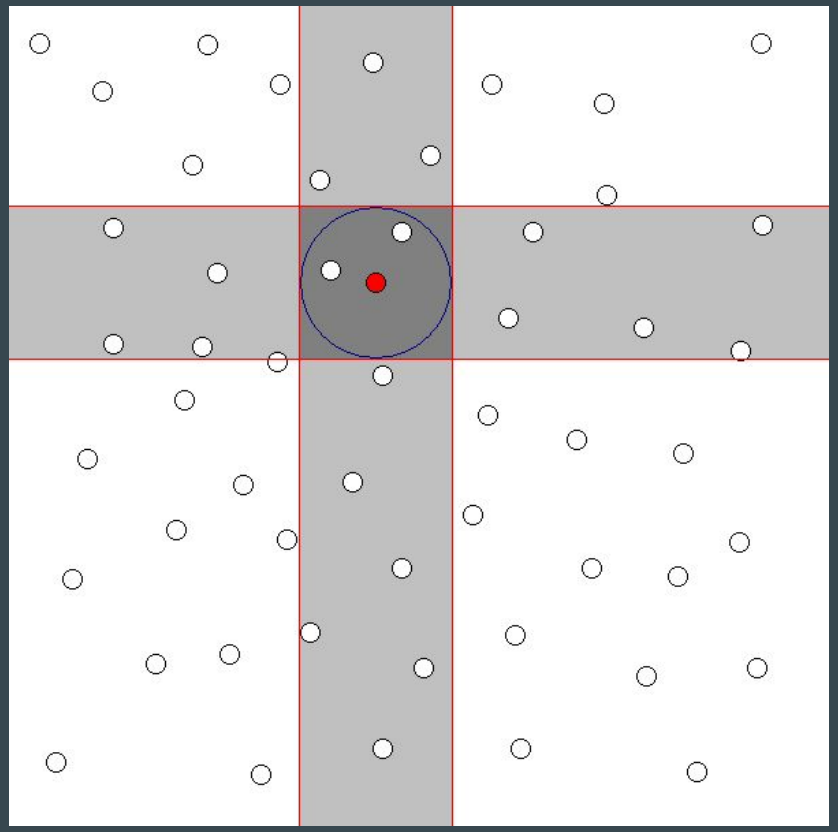
CityFinder

Then, we could filter out elements whose x coordinates are too large or too small



CityFinder

We could then check a much smaller area yet still recover all valid points!



How can we model this in C++?

- We need to be able to efficiently filter cities based off of their x and y coordinates
  - We should be able to only examine cities whose x (or y) coordinate falls in a certain range
- We can do this using iterator ranges!
  - If we keep a mapping between latitude and cities at that latitude, we can iterate through cities within a certain range using `upper_bound` and `lower_bound`

How can we model this in C++

Keep a map from latitude to city name

37.422	Stanford, CA
37.760	San Francisco, CA
39.973	Carmel, IN
43.165	Rochester, NY
43.665	Portland, ME

How can we model this in C++

Keep a map from latitude to city name

37.422	Stanford, CA
37.775	San Francisco, CA
39.973	Carmel, IN
43.165	Rochester, NY
43.665	Portland, ME

How can we model this in C++

Keep a map from latitude to city name

37.422	Stanford, CA
37.775	San Francisco, CA
39.973	Carmel, IN
43.165	Rochester, NY
43.665	Portland, ME

Give me everything from 37.5 to 43.5

`lower_bound(37.5)`

`upper_bound(43.5)`

A Couple of C++ Tidbits

Before we get started with the code, I want to quickly mention a few bits of C++ we'll use

A Couple of C++ Tidbits

`using` allows us to refer to one type using a different name. For example, if I write

```
using FrequencyMap = map<string, int>;
```

Then I can write `FrequencyMap` any time I would have had to write out `map<string, int>`

This makes code easier type and easier to understand.

A Couple of C++ Tidbits

`typedef` does essentially the same thing, but is largely deprecated, as it doesn't work with templating. The syntax is as follows:

```
typedef map<string, int> FrequencyMap;
```

A Couple of C++ Tidbits

Two cities might have the same latitude but different longitudes.

If we stored a simple mapping from latitude values to city names, we might end up overwriting a city which had the same latitude.

A Couple of C++ Tidbits

We can use a **multimap**, which allows a single key to have multiple values.

This means that if we have multiple cities with the same latitude they will be handled appropriately.

Enough Talk!

Time for the code!

(Starting code at http://stanford.edu/~mprecup/cityfinder.cpp)