CSE274 Fall 2012
HW4 – The Closest Starbucks Problem

What is the point of this assignment?
Gain experience with dictionary data structures by designing and implementing a data structure that support fast "nearest neighbor" queries. Experiment with Information Visualization techniques.

Background:
In this project you will be asked to design your own data structure to support the given ADT. You will then build your data structure and test it for speed and accuracy. Finally, you will use it to create a visualization tool to help a company make a decision.

HW04 – Nearest Starbucks

Background info:
Imagine that you need to find the closes Starbucks store to your current location. Google maps makes this very easy – Simply searching for your current address + "starbucks" will work (http://maps.google.com/maps?q=starbucks%2045056). This kind of problem is called the "nearest neighbor problem." In the nearest neighbor problem we start with a large but seldom changing list of locations (that is, we get the x and y coordinates of all Starbucks locations), and build a data structure. This data structure's goal is to make it very efficient to find the closest Starbucks to a given pair of x and y coordinates. In this assignment you will design a data structure that solves the nearest-neighbor problem.

Your data structure will support two operations:

1. void build(entry* c, int n) – You must read in Starbucks_2006.csv, and for each line of the file, create an Entry. These Entries should be put in an array. c is a pointer to this array of "entry" objects, and n is the length of c. The entry class has three member variables: identifier (string), x (the x coordinate, which is between 0 and 1) and y (the y coordinate, which is between 0 and 1). Your build function is expected to store the data and build whatever data structure you want using that data. The running time of build is not important, as long as it finishes in a reasonable period of time (say less than 10 minutes).
2. entry* getNearest(double x, double y) – There is probably not a Starbucks location at position (x,y), so your job is to find the nearest location to it, and return a pointer to the appropriate entry in your data structure. Don't forget that the formula for distance is sqrt((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2)).

100 pts: **HW04 Phase01**
Due MONDAY, October 22 by noon

In this phase you will develop your own design using the "Design Idea Worksheet" and "Design Idea Summary" provided in the starter files. Note that a design based on search trees is fairly natural for this assignment, **but it is not required.** You may use any design or any data structure that you wish. You may work with a group on Phase 1, though each individual must submit which design they are using individually.

Steps:

1) Note: You **MUST** download the Starbucks.h file, and look at the grading criteria for Phase 02, or else you will do badly on this part! The "quality" specifications for your design are contained below, and the interface that your data structure must support is in Starbucks.h
2) Bring hard copies of your "Design Idea Worksheet(s)" and "Design Idea Summary" to class on October 23 for peer evaluation and group activity.

Rubric:

- 50 pts: Thorough and accurate evaluations of your design ideas. This means your "algorithm descriptions" must be correct, and running times correct.
- 25 pts: Thorough consideration of the space of reasonable designs. This means you should have at least two well-written design ideas based on different data structures.
- 10 pts: Address the trade-off between speed and accuracy: For each design, make sure you indicate how accurate the design should be.
- 15 pts: Quality of final design selected. These points are awarded for doing a good job on the "Design Idea Summary," and also for high quality writing, thoroughness, and detail-minded-ness demonstrated in your selected design.

100pts: **HW04 Phase02 – Due MONDAY, Oct. 29, by noon**

In this phase you will implement your design from the previous phase. **Note: You are allowed to change your design, but if you substantially alter your design you will receive a 10% deduction**.

Steps:

1) Create a new repo for this project. **Your repo must be names HW04-uniqueid**. So mine would be called HW04-brinkmwj. If you end up messing up your respository, each new repository should have a number tacked on the end. So , for example, HW04-brinkmwj-2 would be my second try, HW04-brinkmwj-3 would be my 3rd try, and so on. We will grade the highest numbered repository if there is more than one.
2) Download the starter code (Starbucks.h) and the data (Starbucks_2006.csv) from the site. **Do NOT modify Starbucks.h**.
3) Create your own Starbucks class (mine was called brinkmwjStarbucks, and it was implemented in brinkmwjStarbucks.h and brinkmwjStarbucks.cpp) which inherits from the Starbucks class. Note, your class declaration should look something like this: "
   ```
   class brinkmwjStarbucks : public Starbucks {
           /* member variables and methods declared here */
   };
   ```
4) Implement all methods of your Starbucks class. Don't forget the constructor and destructor, as well as the methods inherited from Starbucks.
5) Write a HW04App class that loads the data file, and then creates an instance of your uniqueidStarbucks class, and tests all the methods. **In this phase you do not need to do any graphics**. This means you can do most of your coding in the setup method because all you have to do is setup the data structure, and then write some code to convince yourself that it is working.

Basic goals: [90 pts]

- 40 pts: Project compiles, runs, produces intelligible result. (The entry* that is returned from the query is actually a valid entry from the original input)
- 40 pts: Your getNearest method must do one of the following (but it doesn't have to do both)
  - A) Always returns a result that is within 25% for getNearest, but may be slow. In other words, if the closest Starbucks is distance D away, your getNearest always returns a Starbucks that is at most 1.25*D away.
  - B) Runs in time asymptotically faster than O(N)
- 10 pts: Do both A and B from the previous list
-

Stretch goals: [10 pts]
There is an unavoidable real-life tradeoff between FAST and ACCURATE. For the stretch goal I will be ranking everyone's Starbucks class based on both speed and accuracy.
For ACCURACY: I will call getNearest several times. I will total up the total distance from the point you give me to the input coordinates. The student with the lowest error percentage is best.
For SPEED: I will call getNearest many many times, and time how long it takes. The speed of build() will be used as a tie-breaker.

Both quality measures will be based on the following scale:

- Top 5%: 5 pts
- Top 10%: 4 pts
- Top 25%: 3 pts
- Top 50%: 2 pts
- Works: 1 pt

100pts: **HW04 Phase02 – Due MONDAY, Nov. 5, by noon**

In this phase you will create a visualization using your data structure, and the additional data files, Census_2000.csv and Census_2010.csv

Basic goals: Do whichever of the following you prefer, totaling up to 100 pts.

- A) 20: Draw a map of the continental US, with markers to represent all Starbucks locations.
- B) 10: Let the user click anywhere on the map, highlight the nearest Starbucks, so we can visually tell which one it is.
- C) 20: Make a visual representation of which points on the map correspond to each Starbucks location. For example, you could assign a random color to each Starbucks, and then for each pixel of the map, color it the color of the closest Starbucks for that pixel.
- D) 30: Let the user zoom in and zoom out. In dense urban areas like LA you won't be able to see all the Starbucks locations, otherwise.
- E) 30: Using the census data, color each region based on how much the population increased or decreased from 2000 to 2010. This would use the same idea as C, but the color is no longer random. If population of a region near a Starbucks increased a lot, make the region green. If it decreased a lot, make it red. If it is somewhere in between, make the color something in between.
- F) 30: Using the census data, color each region based on population density, as measured by People Per Starbucks. (Similar to E and C)
- G) 10: Using the census data, color each region based on the change in People Per Starbucks. This is kindof a combination of E and F, so you will get the points for E and F as well, for a total of 70.
- H) Invent your own data visualization that might be of interest to Starbucks management, based on the data you have. Points awarded depend on usefulness/interestingness of your resulting visualization.