CS 120: Intro to Computer Programming II

In-Class Activity - 08 Top-Down Design - Day 1

In this Activity, you will be laying out the design for a program. We'll start at the highest level - defining exactly what the program does - and then break it down, in a Top-Down Design style, into smaller and smaller bits.

There is one critical rule for this Activity: **No arguing!** We are going to be doing something which is open-ended, will require some creativity, and were there is no single "right" answer. Thus, you should expect, right from the start, that your group will do some things that you disagree with. I **encourage** polite, thoughtful discussion about alternatives - but in the end, I'm more interesting in the group coming up with **something plausible** than something that's perfect. Let other people have power and decision-making authority!

Activity 1 - Turn in this one

Today, we will be planning out the design for a program that will find optimal routes between cities. Users will give their starting and ending locations, and the program will tell them what roads to use.

The first step is to more carefully design the problem. For instance, will this program only answer one question, and then shut down, or will it allow the user to make many requests? How will the user interact with the program: will it be text-based, or it will include a GUI of some sort? (Or maybe it will be a website? Or maybe a web service, that can be accessed by other programs?[1])

How will the map be displayed? Will the program just print out a list of cities, one after another? Will it draw the route graphically? Something else? **Discuss the alternatives. What are the tradeoffs?**

Turn in, for this Activity, a summary of the decisions that the group made about the basic ideas of how the program will run.

Activity 2 - Turn in this one

Next, we're going to think about the dataset. There is a map, of course - some sort of data which tells us which cities are connected, and how close they are to each other.

Presumably, you'll want to read the map information from a file. Design what that input file will look like. How much information will you store about each link between cities - will you give just the distance, or will you give the time it takes to drive it? Will your map actually be limited to connecting cities, or will you give more fine-grained information, listing things like the junctions between highways, or small towns on the way? Will you include highway names?

Will you include any information about the cities themselves, or is it sufficient to just list the information about the links between them?

Also, you should consider human-readability. For instance, if you choose to list the time between points, should you list it like humans do (`1:37:23`) - which is easy for humans to read and interpret - or should you list it simply as an integer (5843) - which is easier for your program to parse? Both are reasonable answers - which one does your group like the most?

Turn in, for this Activity, a description of how the input file is formatted, including examples of how the data will be listed in the file.

NOTE: It's quite possible, as you work on this Activity, that your design from the previous step was too complex. If you decide to simplify the program, because you've realized that previously you were trying to keep track of too much information, **that's no problem!** Feel free to update your spec as you get more insight into it.

[1] You all don't know how to make dynamic websites, or web services, yet. But it's OK to dream up an idea, and to say, "the details of exactly how this will work is something we'll figure out later."

Activity 3 - Optional

OPTIONAL. Complete this if you have time, and turn it in. If you don't have time, you may report to your TA that you ran out of time.

Now, it's time to start laying out a plan for your code. Start with the highest levels - reading in the map file, interacting with the user, stuff like that. Describe your program as a series of logical steps - and if you need any loops, clearly describe what they do, and how they know whether to keep looping or not.

Once you've done that, split each step into smaller pieces. I'd recommend that you not try to decompose the "find the shortest path" step yet - save that for the next Activity. But, for everything else, break it down into as small pieces as you reasonably can do.

Special Requirement: Don't write any code! Just design *how* the code will work. Pay attention to details, here - you might, for instance, want to detail exactly what data types are sent from one step of the program to the next. What are some likely functions that you will need to write, and what are their parameters and return values? What variables will you be using, and exactly what will their contents be?

If you complete this step as well as you can - if you've broken the program into such small pieces (except for the actual pathfinding algorithm) that the next step would be to write some code, move on to the last step of today's Activity.

Activity 4 - Optional

OPTIONAL. Complete this if you have time, and turn it in. If you don't have time, you may report to your TA that you ran out of time.

The Right Way to solve this problem is with something called Dijkstra's Algorithm, but most of you haven't ever written this, and you probably won't until you take CSC 345.

But that's OK - let's play around with algorithm design anyhow. We know that what you come up with probably will be **much less efficient** than Dijkstra's - but that's just fine.

So, talk it out with your group. If you have to get from A to B, and to do that you'll need to find some sort of path through lots of links connecting the cities together, how might you begin your exploration? Start to work out the details of your algorithm:

- How do you decide what path (or part of a path) you want to explore first?
- What information do you store as you explore, and how does it change (new info, deleting old info, or improving paths) as you find out new things?
- There are probably many paths from A to B; how do you choose between them? And how do you know when you've found the best one possible?