CompSci 260P Winter 2020 Project 1: Dynamic Programming

Due Tuesday, February 4, 6:29PM, although late submissions are allowed.
Note that the due date is 24 hours (plus one minute) prior to your first exam.

Getting Started

Before you begin work on this project, there are a couple of chores you'll need to complete on your VM to get it set up to proceed.

Refreshing your CompSci 260P VM environment

Even if you previously downloaded your CompSci 260P VM, you will probably need to refresh its environment before proceeding with this project. Log into your VM and issue the command `compsci260p version` to see what version of the CompSci 260P environment you currently have stored on your VM. If it does not indicate that project 1 (or a later project) is posted, you need to run `compsci260p refresh`.

If you're unable to get outgoing network access to work on the VM — something that afflicts a handful of students each quarter — then the refresh command won't work, but an alternative approach is to download the latest environment from the link below, then to upload the file to your VM using SCP. (See the Project #0 write-up for more details on using SCP.) Once the file is on your VM, you can run the command **compsci260p refresh_local NAME_OF_ENVIRONMENT_FILE**, replacing **NAME_OF_ENVIRONMENT_FILE** with the name of the file you uploaded; note that you'd need to be in the same directory where the file is when you run the command.

The file is linked from the "public" CompSci 260P page; click this link and enjoy the amazing web design skill that put it together: https://www.ics.uci.edu/~mikes/compsci260p/

Creating your project directory on your CompSci 260P VM

A project template has been created specifically for this project, containing a similar structure to the basic template you saw in Project #0.

Decide on a name for your project directory, then issue the command **compsci260p start YOUR_CHOSEN_PROJECT_NAME project1** to create your new project directory using the project1 template. (For example, if you wanted to call your project directory proj1, you would issue the command `compsci260p start proj1 project1` to create it.) Now you're ready to proceed!

Reviewing related material

Look over the second half of the lecture from January 22 (Lecture 5). If you haven't ever implemented dynamic programming before, I encourage you to try a few simpler ones first. In your textbook, telescope scheduling (12.3) and 0-1 Knapsack (12.6) are good suggestions. Feel free to work with any number of your classmates on this part, including understanding the algorithm from lecture and even implementing anything from chapter 12's reading.

This project is meant to make sure you *really* know how to implement a dynamic programming solution. Even if you never implement dynamic programming again (although you likely will), this is still a skill that will translate to other areas of the field.

Choosing a project partner

You *have the **option*** to work with a second person for this assignment. If you do so, I expect you to work via pair programming. That is, you **may not** split the assignment, such as by having one person implement the dynamic programming while the other person implements the part that finds the journey given the table, and the two are stitched together later. I reserve the right to ask one or both project partners about the implementation and adjust the score accordingly.

Similarly, any academic dishonesty arising from a group will be treated as an offense by both partners.

Both partners must fill out the following survey to register the partnership. It is not enough for one to do so. Be sure to include your UCINetID and your partner's UCINetID. Failure to do so may cause one person to not get credit.

https://docs.google.com/forms/d/e/1FAIpQLSevC1LT2jjopViR1ZJU27fU1IMbaAByVsf2S4RGiMPnbTuXPg/viewform

Requirements

You are required to implement the functions `tsp_dynamic_program` and `costOfJourney` in proj1.cpp. These have the same signature as the corresponding functions in project 0, **except** the input may be larger for `tsp_dynamic_program` than it could have been for `tsp_brute_force`, in part because we expect a more elegant (and faster) solutions. Oh, and the first function has a different name.

For this project, you have a few requirements:

- You must implement the functions in proj1.cpp**.**
- You must use dynamic programming for `tsp_dynamic_programming`. The good news is that we have provided you with a solution in lecture. Of course, there's some space in between a solution "on paper" as in lecture and some implementation details. *That's for you (perhaps with a partner) to figure out*.

- Your program must run in under three minutes on a reasonably modern computer. Test cases that take longer than this to run may be deemed to be incorrect. Note that this means you will need to think a little about efficiency in your program. It also imposes a restriction on me: I cannot give you a very large test case. Here are some sample running times; you **do not** need to match these, nor are these presented as fully optimized. Each of these is from a single run of a single test case that will be used when grading your assignment.
    - n=20, 4016 ms
    - n=21, 8432 ms
    - n=22 18034 ms
    - n=23 39822 ms

- You may assume all inputs are valid; for example, in `costOfJourney`, the second parameter will always be a permutation of the unsigned ints in the range of [0, n-1], where *n* is the number of vertices in the given graph. You do not need to do bounds checking or the like.

You *may* use standard libraries as appropriate, unless there is one that makes solving this problem trivial. I am unaware of any such part of the library.

You are **explicitly permitted** to use C++ standard library container classes (std::vector, std::set, etc). You are welcome to ask anything you want about these libraries, or to look up material about them online. Information about how to use an explicitly-permitted library may always be shared among classmates, but refrain from telling one another how you solved a problem in the assignment with them. For example, answering "how do I check if an element is in a std::set?" is great and encouraged, while answering "what did you use std::set for in your project?" is not.

A good reference for the STL container classes (such as those listed above, including std::map) is http://www.cplusplus.com/reference/map/map/ .

If you would like to reuse some or all of *your code* (not that of someone else) from project 0 for part of this project, you are welcome to do so. If you have a partnership, you may take code from either or both partners' versions of project 0, but not from anyone else's.

Remember that the purpose of this project isn't to *find* an implementation of TSP, but rather to code it yourself. Submitting work that isn't yours (for any reason) is a decidedly bad idea and one of the very few ways to do poorly in this class. If I find that you submitted code you found online, your grade will be worse than if you hadn't turned in this project at all.

Deliverables

After using the gather script in your project directory to gather up your C++ source and header files into a single **project1.tar.gz** file (as you did in Project #0), submit that file (and only that file) to Checkmate. Refer back to Project #0 if you need instructions on how to do that. This time, it should give you the correct file name, insert innocent-looking face emoji here.

You will submit your project via Checkmate. Keep in mind that that you're responsible for submitting the version of the project that you want graded. We won't regrade a project simply because you submitted the wrong version accidentally. (It's not a bad idea to look at the contents of your tarball before submitting it; see Project #0 for instructions on how to do that.)

Which partner should submit?
If you are working as a partnership, **exactly one** of you should submit the project. Do not forget to submit it, and also, I do not want to have the situation where both partners submit it. If two people both submit it, there may be a grade penalty assessed.

Can I submit after the deadline?
Yes, it is possible, subject to the late work policy for this course, which is described in the section titled Late work in the course reference and syllabus. The short story is, the longer after the deadline, the less credit you get, and after 99 hours post-deadline, no credit. Note that the first exam is less than 99 hours post-deadline and you cannot move the exam in order to make time to do this project.

Grading
The grade that will be listed in Canvas is your "raw score" based solely on correctness : we will run some number of test cases for your code and, based on how many and which ones you get correct, you will earn some number of points between 0 and 10 (inclusvive, and might not be integer-valued). Each is worth some number of points and is graded based on whether or not your code correctly determines if the puzzle has a solution, and if so, what it is. If it is determined that your program does not make an attempt to solve the problem at hand, you will not get these points, regardless of the result from testing. The tests will look a lot like the tests in your Google Test starting directory for this assignment; if you pass those, you're off to a good start, but it's not a guarantee.

The grade posted on Canvas does not include late penalties.

If we review your code and find your style to be sufficiently bad, we reserve the right to deduct points based on this, proportional to how bad the style is. If we do so, we will alert you to both the penalty and the reason. Here are some guidelines to follow when submitting your code:

- Include a reasonable level of comments. Do not comment every line, but do not omit comments either. A decent guideline is that if you were asked to explain your code six months from now, your comments should guide you to be able to do so.

- Use meaningful variable names where appropriate. Loop counters need not have a long name, but if you declare a std::set, give it a name reflecting what is in the set, not "mySet."

- Use proper scoping for variables. Avoid global variables except in rare circumstances. Pass parameters appropriately.

- Indent appropriately. While C++ does not have Python's indentation requirements for writing usable code, the guidelines of *readable* code are an issue here.

- Avoid vulgarity in your code and comments. Variable names, output statements, and the like should not make reference to topics that are not discussed in polite company.

- Remove debug output as appropriate; rather than commenting it out, delete it if it is unnecessary. If you think you might want to return to those debug statements, enclose them instead. Write something like this as a global variable (this is a case where such is acceptable):

  ```
  const bool DEBUG_OUT = true;
  ```

  Then, when you need a debug statement, enclose it:

  ```
  if( DEBUG_OUT )
  {
        std::cout << "Set has been successfully declared!" << std::endl;
  }
  ```

  When you want to remove the debug output, you can change the declaration to set the variable to false. You may wish to have multiple variables to control debug in various functions, either globally or having them local to functions.

  Similarly, if you keep your debug output lines in the file you turn in, having meaningful output statements is better than "code is here!" or "aaaaa." This is also true if you are going to ask someone (like your professor, for example) for help with debugging.