CompSci 260P Spring 2023 Project 3: More Dynamic Programming
Due May 1, 2023, at 6:00 PM, although late submissions are allowed.
Note that the maximum submission time is also the date of your first exam.
Plan accordingly.

Requirements
You are required to implement the functions tsp_dynamic_program and costOfJourney in tspdynp.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) solution. Oh, and the first function has a different name.

For this project, you have a few requirements:

- You must implement the functions in tspdynp.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 to figure out.
- The journey you return from tsp_dynamic_program must begin at vertex zero. Code to adjust the journey accordingly is commented out in the provided file.
- Your program must run in under three minutes on the grading computer. See previous projects for the caveat to this; I suggest you do not cut it close if you have a newer 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 may 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/](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.

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.