Title page

CS404, Fall Semester 2016: Find the Sneaky Path

I understand and have adhered to the rules regarding student conduct. In particular, any and all material,

including algorithms and programs, have been produced and written by myself. Any outside sources that I

have consulted are free, publicly available, and have been appropriately cited. I understand that a violation

of the code of conduct will result in a zero (0) for this assignment, and that the situation will be discussed

and forwarded to the Academic Dean of the School for any follow up action. It could result in being

expelled from the university.

//Your Name// //This Date//

Please print and sign this page. Subsequently scan it back in and attach it to your report.

Introduction (5%) Section in which you state the problem in your own words or paraphrased from this description. This

makes the report self contained so that you (and others, like employers) can read the rest with ease, even next

semester/year.

Design and Analysis of the algorithm(s) (25%) A section in which you present your algorithms with supporting data

structures. Provide convincing arguments that your algorithms finds the correct shortest paths, finds the correct

loading on each link, and finds the correct SneakyPath for the given a and b. This includes an analysis in terms as best-case and worst-case time complexities as well as space complexities. Make sure to mention the key-and-basic

operation you are using. The analysis is expected to have appropriate rigor and communicated clearly, so that a student

who is currently in CS303 is able to follow your derivation. In your narrative, explain any design and implementation

decisions you made and why you believe these were good choices. In particular, highlight any special or innovative

features and focus on the performance impact of your decisions. Describe any modifications that you had to make to

any standard algorithm as presented in the textbook, in class, or those you found elsewhere (give proper citations).

Highlight any key insight or neat design feature that is particularly helpful for efficiency in this project, and those that

were particularly enlightening to you.

Implementation and testing of your algorithm (25%) First, you need to tell us what language you use to implement your

algorithms, and why you have decided on this particular language. Also, in this section you provide evidence that

your program is running and gives correct results, probably for small test cases you create yourself by hand. If you

needed to make significant changes to the original algorithm or original program, then you report these in this section,

and you report which steps you took to convince yourself (as well as the TA and me) why your program is a correct

implementation of the algorithm, and report on the various test data you created. We expect the code to be transparent

and ’mean and lean’. A listing of all your programs (both .h and .cpp files) are in an appendix. Elaborate when

appropriate. For instance: if there are two closely related data structures, then how did you decide between them?

How and why did you break ties? Why did you rely on routines from the STL with unknown performance, rather

than writing your own and understand it’s performance? Highlight any key insight or neat design or implementation

feature that is particularly helpful for the efficiency in this project, and those that were particularly enlightening to

you.

Validate your pre-implementation analysis with a timing study of your program (15%) Perform a timing study of your

implementation and show that this conforms with your (pre-implementation) time complexity study. In other words,

measure the run times of your programs to experimentally verify your analytical findings. This is preferably done in

one or more tables, where you present, for each input, the value of n, the predicted time complexity of your algorithm

and the measured time consumption of your programs. Now is the time for the analysis and interpretation: Are these

timings consistent with the predicted times? Can you daw any conclusions from these? If not, you may need to

generate additional test cases. Can you draw conclusions if you also include the smaller test cases? Based upon your

work (you may need to do additional runs): how long would your algorithm/program run if you double the input size?

You may need to copy this information from the output as generated by your program. Include in an appendix the

(unedited) outputs as generated by your program(s) for the required test cases that will be provided.

Epilogue (10%) In this section, you reflect back on the design and data structure decisions you made. How did you handle

any unforeseen situations and any ’newly discovered’ problems? Now that you have done the project, what have you

learned? If you have the opportunity to do this project again, perhaps with additional requirements, what would you

do the same, and what would you do differently? What are the lasting lessons you have learned? What are the lessons

for future students?

Written communication skills (10%) Organization, readability, structure, style and presentation of the report. Use correct

English (grammar, syntax, spelling, and so on) and use section headings that will make it easy for us to find the

outcomes we are looking for.

Appendix A: Program Listing (5%) Here provide a ‘Program Listing’. Include both .h and .cpp files. Programs should

be documented according the standards from CS201/CS201L/CS303. State which compiler and which machine was

used to produce the results. We expect your program to be readable and documented internally: identification block;

Internal documentation, proper indentation, meaningful variables, . . .

Appendix B: Output Provide outputs exactly as generated by the programs you include in appendix A.