CS 1240 Programming Assignment 1: Spring 2025

Your name(s) (up to two):

Collaborators: (You shouldn't have any collaborators but the up-to-two of you, but tell us if you did.)

No. of late days used on previous psets:
No. of late days used after including this pset:

Homework is due Friday Feb. 28 at 11:59pm ET. You are allowed up to **twelve** late days through the semester, but the number of late days you take on each assignment must be a nonnegative integer at most **two**.

Overview: The purpose of this assignment is to experience some of the problems involved with implementing an algorithm (in this case, a minimum spanning tree algorithm) in practice. As an added benefit, we will explore how minimum spanning trees behave in random graphs.

Assignment: You may work in groups of two, or by yourself. Both partners will receive the same grade and turn in a single joint report. (You should submit a single copy of the report, with both of you listed as submitters, in Gradescope.)

We recommend using a common programming language such as Python, Java, C, or C++. If you use a more obscure language, that is fine, but if there are errors that are correspondingly harder to find it may cause your grade to be lower. In previous years the assignment was designed to be very difficult to complete in Python due to runtime concerns – it has been updated this year so this is no longer the case, so any advice you may have received to learn C++ for this assignment should be taken with a grain of salt.

For any algorithms or data structures taught in CS 1240 that you wish to use, you must write your own code. For instance, you may not use external libraries for heaps, priority queues, or finding MSTs outright, but you may use external libraries for, e.g., binary search trees, hashing (e.g. built-in Python dictionaries), or routine operations on arrays.

For most of the following cases, we will be considering *complete, undirected* graphs. A graph with n vertices is complete if all $\binom{n}{2}$ pairs of vertices are edges in the graph. The hypercube graph is the only exception – please pay special attention to that description.

Consider the following types of graphs:

- Complete graphs on n vertices, where the weight of each edge is a real number chosen uniformly at random on $[0, 1]$.

- "Hypercube" graphs on n vertices, where (a, b) is an edge iff $|a - b| = 2^i$ for some i, and the weight of each edge is a real number chosen uniformly at random on $[0, 1]$. (This does not exactly match the true definition of a hypercube graph, but it is the definition we will work with for simplicity)

- Complete graphs on n vertices, where the vertices are points chosen uniformly at random inside the unit square. (That is, the points are (x, y), with x and y each a real number chosen uniformly at random from $[0, 1]$.) The weight of an edge is just the Euclidean distance between its endpoints.

- Complete graphs on n vertices, where the vertices are points chosen uniformly at random inside the unit cube (3 dimensions) and hypercube (4 dimensions). As with the unit square case above, the

weight of an edge is just the Euclidean distance between its endpoints.

Your goal in this Programming Assignment is to determine, in each of the three cases above, how the expected (average) weight of the minimum spanning tree (not an edge, the whole MST) grows as a function of n.

We give further guidelines on how to get to this goal in the rest of this handout. This will require implementing an MST algorithm, as well as procedures that generate the appropriate random graphs. You may implement any MST algorithm (or algorithms!) you wish; however, we suggest you choose carefully.

For each type of graph, you must choose several values of n to test. For each value of n, you must run your code on several randomly chosen instances of the same size n, and compute the average value for your runs. Plot your values vs. n, and interpret your results by giving a simple function $f(n)$ that describes your plot. For example, your answer might be $f(n) = \log n$, $f(n) = 1.5\sqrt{n}$, or $f(n) = \frac{2n}{\log n}$. Try to make your answer as accurate as possible; this includes determining the constant factors as well as you can. On the other hand, please try to make sure your answer seems reasonable.

Code setup:

So that we may test your code ourselves as necessary, we prefer that your code accepts the following command line form:

./randmst 0 numpoints numtrials dimension

Accepting that command line form is optional. If your code doesn't and we need to test your code, we may need to contact you.

The flag 0 is meant to provide you some flexibility; you may use other values for your own testing, debugging, or extensions. The value numpoints is n, the number of points; the value numtrials is the number of runs to be done; the value dimension gives the dimension. (Use dimension 0 for the complete graph with randomly distributed weights, and dimension 1 for the hypercube graph with randomly distributed weights; use dimensions 2, 3, 4 for the complete graphs on points in 2D, 3D, and 4D space, respectively; Notice that points in 0 or 1 dimensions are not particularly interesting, so we are reusing the 0 and 1 labels) The output for the above command line should be the following:

average numpoints numtrials dimension

where average is the average minimum spanning tree weight over the trials.

It is convenient for us in grading to be able to run the programs without any special per-student attention. The following three instructions make that easier for us, but don't spend more than a few minutes trying to satisfy them—we can contact you to ask about seeing the code run on your machine if necessary.

- If possible, for compatibility reasons, the code should run on a Unix/Linux system, even if you code on another system.

- We expect the code as described above in its own file(s) separate from your results/writeup.

- The code should compile with make; no instructions for humans. That is, the command "make randmst" should produce an executable from your directory. You may need to read up on makefiles to make this happen.

Note: you should test your program – design tests to make sure your program is working, for example by checking it on some small examples (or find other tests of your choosing). You don't need to put your tests in your writeup; that is for you.

What to hand in: Besides *submitting a copy of the code you created*, your group should hand in a single well organized and clearly written report describing your results. For the first part of the assignment, this report must contain the following quantitative results (for each graph type):

- A table listing the average tree size for several values of n. (A *graph* is insufficient, although you can have that too; we need to see the actual numbers.)

- A description of your guess for the function $f(n)$.

For all *complete graphs*, run your program for $n = 128$; 256; 512; 1024; 2048; 4096; 8192; 16384; 32768; and larger values, if your program runs fast enough. For the hypercube graph (dimension 1), run your program for $n = 128$; 256; 512; 1024; 2048; 4096; 8192; 16384; 32768; 65536; 131072; 262144 (Providing results only for smaller n will hurt your score on the assignment.) Run each value of n at least five times and take the average. (Make sure your experiments use independently generated randomness!)

In addition, you are expected to discuss your experiments in more depth. This discussion should reflect what you have learned from this assignment. You should, at a minimum, discuss the asymptotic runtime of your algorithm(s) and the correctness of any modifications from the standard algorithms presented in class; otherwise, the actual issues you choose to discuss are up to you. Here are some possible suggestions for the second part:

- Which algorithm did you use, and why?

- Are the growth rates (the $f(n)$) surprising? Can you come up with an explanation for them?

- How long does it take your algorithm to run? Does this make sense? Do you notice things like the cache size of your computer having an effect?

- Did you have any interesting experiences with the random number generator? Do you trust it?

Your grade will be based primarily on the correctness of your program and your discussion of the experiments. Other considerations will include the size of n your program can handle. Please do a careful job of solid writing in your writeup. Length will not earn you a higher grade, but clear descriptions of what you did, why you did it, and what you learned by doing it will go far.

Hints:

To handle large n, you may want to consider simplifying the graph. For example, for the graphs in this assignment, the minimum spanning tree is extremely unlikely to use any edge of weight greater than $k(n)$, for some function $k(n)$. We can estimate $k(n)$ using small values of n, and then try to throw away edges of weight larger than $k(n)$ as we increase the input size. Notice that throwing away too many edges may cause problems. Why will throwing away edges in this manner never lead to a situation where the program returns the wrong tree?

You may invent any other techniques you like, as long as they give the same results as a non-optimized program. Be sure to explain any techniques you use as part of your discussion and attempt to justify why they should give the same results as a non-optimized program!