Link to this file (anyone with the link can view):

https://docs.google.com/document/d/1OZ6eEb9E_SaIrORRfE5xZ3uw-n1cfCyDkVPuAlvt90M/edit?usp=sharing

ABK advice for W18 midterm preparation (and, taking the midterm itself):

1. About understanding given "Algorithm Design" problems (likely, Q3 and Q4 of the MT)
    a. Look at the small example(s) usually given with the problem. It's there to make sure that you are clear about what the desired algorithm is supposed to do.
    b. Understand the properties that your algorithm MUST -- or, MUST NOT -- have. E.g., questions often say things like: "No credit will be given for an O(n^2) solution", or "The solution to Problem 2 on HW2 will NOT work for this!", etc. Please don't ignore these statements.
2. About getting started on "description and pseudocode"
    a. Your English description will map (hopefully) to pseudocode, so try to match up these parts of your answer. Mappable English (in my personal view) has the flavor of: "The main idea of my (the) algorithm is to … My algorithm first does …. Then, for each … my algorithm does … Then, … Finally, the desired output … is obtained by … Because … the runtime is O(...)"
    b. A principle of problem-solving that applies to algorithm design, and that may be helpful if you are stuck: what's the "penultimate step" (== next-to-last step)? If you are delivering a sum, an index, a tree, a path, a cycle, a … -- how did you compute this last thing that you are delivering? From what did you compute it, in what structure did you find the "from what", ...?
    c. This also brings up a "(keep, begin with) the end in mind" precept: if you know that you're going to deliver a matrix of vi-to-vj shortest-path costs (for example), then you know you'll be filling in (the 'moral equivalent of') a matrix of |V|^2 entries. You can then work backward: How did I fill in a single entry? Or, how did I fill in a row of entries? Etc. Remember, these are "puzzles" - solutions exist and are findable using the tools that we know that you know! (See below.)
    d. Lecture 1: notation, terminology are important (you and your reader must be on the same wavelength!)
        i. What notation, terminology are given and "established"? n, V, |V|, i, j, …?
        ii. Staying in this given/established framework is likely to make life easier for everyone!
3. Did you attend the MT review?
    a. Try to look at the slides and podcast.
    b. Will try to dig up a small practice problems document for those interested. Team will post ...
4. Go over the posted Quizzes and Midterms from 2014 and 2015 and 2016.
    a. Please understand and be able to do recent years' problems.

    i. NOTE: Some “Quiz 0” or “Quiz 1” questions were really intended for CSE 21 or “mathematical maturity” review. Don’t worry about such questions; we will be “in proper scope” with MT questions.
  b. The midterm structure will be similar to that of previous midterms (~4 problems, 40 or 50 points: Short-Answer (~10 or 20), “Mechanical Execution” (~10), Design/Analyze I (D/Q) (~10), Design/Analyze II (graphs/SPs) (~10).
  c. Review of quizzes and TTK should help with Short-Answer (~10 or 20)
  d. Fair game for Design/Analyze obviously includes DFS, D/Q, Dijkstra
5. Start your one-sheet notes (aka “cheat sheet”) **early**.
  a. Preparing it is in itself very helpful in constructing a “mental map” of the course material.
  b. Start by writing down your own takeaways from each of the six lectures so far. (And, after Tuesday, from Lecture 7 as well -- see Slide 6 and Slide 13 in the Tablet version for the “takeaways / key points” that we discussed in class.)
  c. IMO, items of “advice” in this doc suggest things that could be usefully written down in your “cheat sheet”...
6. Prepare for specific kinds of questions
  a. Execute SCC-finding, Dijkstra, and Bellman-Ford on small graphs. (You know that at least one full question (#2) on the midterm will be such “Mechanical Execution” (~10 points).)
  b. Remember conventions / notations established in class.
    i. Break all ties in lexicographic order.
    ii. SPs (Dijkstra): “label”, “set R”, …
    iii. SPs (Bellman-Ford): l_j^(k) means what? **// likely not on MT**
    iv. Edge (u,v) in a directed graph: which way does the edge go?
    v. Etc.
  c. Give big-O analysis of small pseudocode snippet (e.g., by writing down the applicable recurrence for T(n), applying Master Theorem, etc.)
  d. Note about recurrences: to the extent that this comes up, knowing Master Theorem and how to apply it should suffice.
7. A comment on “Algorithm Design” questions
  a. The “leap” from known techniques or solutions to the “new” problem on the MT will be much smaller than what you’ve had to contend with on HW questions.
  b. The good news is that MT questions are “puzzles”, not “problems”: solutions are known to exist that satisfy given requirements, they are known to be solvable using the tools that you have, by leveraging a solution approach that you’ve already seen, etc.
  c. Also, “use all the information given” - e.g., if there is a big-O time bound given, that limits some design options, perhaps...
    i. Example: “Design a linear-time algorithm to …” → you can’t sort the input as the first step of your algorithm
    ii. (N.B.: There is an old problem-solving precept: “If you haven’t used all the information given, something might be fishy with your solution…”)

8. Other
    a. Make sure that you know and understand the TTK page http://vlsicad.ucsd.edu/courses/cse101-w16/ttk/index.html Item #'s 1-23 especially.
    b. Understand distinctions
        i. connected vs. strongly connected
        ii. meaning of three cases of Master Theorem
        iii. instances for which Dijkstra does not return correct shortest path costs from source
        iv. when Bellman-Ford "can be stopped early" **// not on W18 MT**
        v. etc.
    c. Understand respective time complexities of DFS, BFS, Dijkstra, Bellman-Ford discussed in textbook/lecture.
        i. As mentioned, these will also generally be given on the cover page of the MT (similar to previous MTs). We will discuss on Tuesday that the given time complexity for Dijkstra will assume a binary heap PQ implementation (see page 125 of DPV). This being said, please do understand these time complexities.

If you've done the HWs + PA (= good understanding of explore, dfs), and if you've done the above, you should be in good shape for the MT !!!