Homework 5: Graphs and Loose Ends

Handed out Monday, November 28. Due at the start of class Thursday, December 8–NO LATE ACCEPTED.

Problem 1. This problem is designed to refresh your memory on some of the properties of AVL trees.

(i) Please draw *all* valid AVL trees containing $n \leq 4$ nodes.

(ii) This was on the exam, but most students didn't get to it... Recall that $N(h)$ is defined to be the minumum number of nodes in an AVL tree of height h, and that

$$N(h) = \begin{cases} 0 & \text{if } h = 0 \\ 1 & \text{if } h = 1 \\ N(h-1) + N(h-2) + 1 & \text{otherwise} \end{cases}$$

Prove by induction that $\forall h, N(h) = F_{h+2} - 1$.

(iii) The following is a corollary of Binet's Formula:

$$F_i > \frac{1}{\sqrt{5}}(\phi^i - 1) > \frac{\phi^i}{\sqrt{5}} - 1$$

(This is because $\forall i, \hat{\phi}^i < 1$.) Using this corollary and Part (ii) above, show that the worst-case height of an AVL tree with n nodes is $O(\log_\phi n)$ (which implies that any search/insert/delete operation will run in $O(n \log n)$.

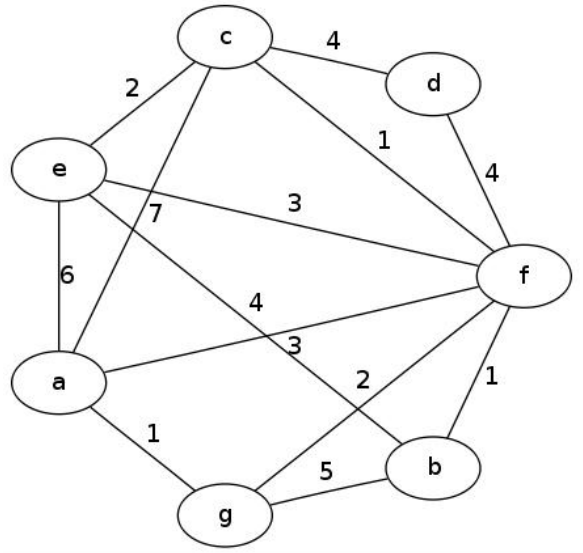
Problem 2. For each part below, draw a graph G that satisfies the given criteria. Your graphs should be *connected*, *bidirectional*, and should only consist of a *reasonably small* number of nodes.

(i) G has an Eulerian cycle, but not a Hamiltonian cycle.

(ii) G has a Hamiltonian cycle, but not an Eulerian cycle.

(iii) G has both an Eulerian and a Hamiltonian cycle.

Problem 3. Use any algorithm you would like to generate an MST for the following graph. Explain which algorithm you used and the steps you followed to build the MST, so that we can give partial credit if necessary!



Problem 4. The psedocode for Tree Growing is given below:

```
def tree_grow(start):
  reached = {}
  fringe = {start}
  while fringe is non-empty:
    curr = "get next node" from fringe
    fringe = fringe - {curr}
    "process" curr
    reached = reached U {curr}
    for all children c of curr:
      if c "is interesting" and c is not in reached:
        fringe = fringe U {c}
```

For each of the following algorithms, explain how to implement the quoted primatives in the tree-growing algorithm. The first one is done for you:

- Depth-First Search (print the current node at each step in the traversal)
  - "get next node" - the fringe is a stack. Getting the next node is just peeking at the top of the stack, assuming that node has not been reached already.
  - "process" - here, just print the current node.
  - "is interesting" - children here are always interesting, as long as they are not already reached.
- Breadth-First Search (print the current node at each step in the traversal)
  - "get next node" -
  - "process" -
  - "is interesting" -

- Prim's algorithm for MST
  - "get next node" -
  - "process" -
  - "is interesting" -
- Dijkstra's algorithm for Shortest Path
  - "get next node" -
  - "process" -
  - "is interesting" -

Problem 5. Consider the following problem from AI, known as *best-first search*: Suppose you are given a graph G, a start node s, and an end node f. You want to construct a relatively efficient path from s to f, if one exists, but you do not have an exact weight function for edges in the graph. Instead, you are given a heuristic function $h : V \to R^+$ that represents roughly the "goodness" of a particular node in terms of how close it is to the solution. In order to construct your path, you need to perform a search of nodes in the graph, starting from s, and you want to explore nodes in order, from highest $h(v)$ to lowest. (This sort of search algorithm is often used in planning, where the nodes in the graph represent states, and are not always known at the beginning, but may be generated based on the state you are currently in by a series of rules). Please describe how you would implement the following tree-growing "primitives" to perform this kind of search:

- "get next node" -
- "process" -
- "is interesting" -

Problem 6. This is kind of a miscellenous problem and does not fit the general theme of this homework, but it is important nonetheless! Use the integration method to find positive constants c_1 and c_2 such that

$$c_1 \leq \sum_{k=1}^{\infty} \frac{1}{k^4} \leq c_2$$

Note: Giving values of c_1 and c_2 without any proof will earn zero credit. Also note that the value of the integrand decreases as k increases.

Hint: check the Wikipedia page linked to the course resource page, and feel free to search a table of integrals if you forget that the antiderivative of k^{-4} is $-(k^{-3})/3$ plus a constant, of course.

Extra hint from Greg: this problem is *hard*. Directly applying the integration method (as defined on Wikipedia) will make you a little bit unhappy (try it to find out why). But, one can resolve this by cleverly splitting the summation into two pieces...