Collaboration Policy: You are encouraged to collaborate with up to 4 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: list your collaborators

Sources: Cormen, et al, Introduction to Algorithms; Adams, Douglas, Hitchhiker's Guide to the Galaxy

PROBLEM 1 *Asymptotics*

1. Consider the following functions, $f(n) = n^{1.5}$ and $g(n) = n(\log n)^2$. Which grows more quickly? (That is, which would be a "worse" time-complexity.) Express your answer in terms of one of the order-classes we've studied. If you can use little-omega ω or little-oh o, use that. If not, use big-Oh O or big-Omega Ω. Your answer will be of the form $f(n) = \Theta(g(n))$ but with something other than Big-Theta Θ. Explain your answer with a short proof; you may use any definitions from the course slides.

Solution:

PROBLEM 2 *Trick or Treat: Maximum Candy*

October 31st is just around the corner and Prof. Bloomfield's children are making big plans for trick-or-treating in their neighborhood. In an attempt to limit the amount of candy his children can collect, Prof. Bloomfield – being a mean father – told them that they can only go 2 blocks away in any direction. The children will be starting at their house, which happens to be next to an intersection, meaning that can choose one of 4 street segments to start on. Each street segment has a number of houses that provide candy on it. Given a map of the neighborhood, what will Prof. Bloomfield's dental bill be?

Rather, how much candy can each child collect?

The map can be visualized as a graph containing nodes (intersections) and edges (street segments). Edge weights represent the number of houses that provide candy on a given street segment. In the figure to the right, if the children start at node S (in the middle of the graph), there are three streets (edges) that they cannot travel to, as they are more than 2 blocks away: J-F, G-H, and E-I. The total candy they can collect is the sum of the other edges, or 44.

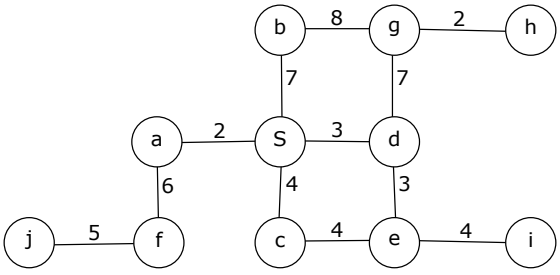

Figure 1: Neighborhood roads

Given a weighted, connected, unidirectional graph $G = (V, E)$, a designated start vertex s, and a parameter n (the maximum number of blocks that can be traveled), create an algorithm that finds the amount of candy that the

children can collect. Clearly describe your algorithm. Analyze and state the time complexity of your algorithm using asymptotic notation. *Note that your algorithm must work for* **any** *graph, not just the one shown here.*

Solution:

PROBLEM 3 *Trick or Treat: Parental Supervision*

Given the number of children that will be trick-or-treating in Prof. Bloomfield's neighborhood, he really thinks parents should be watching the streets. When a parent is present at a given intersection they can see each street segment that connects to that intersection.

Use the same graph representation as in the previous problem. Note that while the graph above is orthogonal (all the streets are on a grid, the provided graph may not be – there may be any number of roads connecting to an intersection, a block of three streets (edges), etc. Create an algorithm that finds the minimal number of parents needed such that each street segment has a parent stationed at one of the connecting intersections. In the graph above, the answer is 4 (nodes E, F, G, and S). Clearly describe your algorithm. Analyze and state the time complexity of your algorithm using asymptotic notation. *Note that your algorithm must work for* **any** *graph, not just the one shown above.*

Solution: