1 Trie Your Best

(a) What strings are stored in the trie below? Now insert the strings `indent`, `inches`, and `trie` into the trie.

(b) What is the runtime to find out if a given string is in the tree? What is the runtime to add a string to the tree? Describe your answers in terms of $N$, the number of words in the trie. You may assume the max length of any word in the trie is a constant.

(c) Extra: How could you modify a trie so that you can efficiently determine the number of words with a specific prefix in the trie? Describe the runtime of your solution.
2 A Tree Takes on Graphs

Your friend at Stanford has made some statements about graphs, but you believe they are all false. Provide counterexamples to each of the statements below:

(a) "Every graph has one unique MST."

(b) "No matter what heuristic you use, A* search will always find the correct shortest path."

(c) "If you add a constant factor to each edge in a graph, Dijkstra’s algorithm will return the same shortest paths tree."
3 Graph Algorithm Design

(a) An undirected graph is said to be bipartite if all of its vertices can be divided into two disjoint sets $U$ and $V$ such that every edge connects an item in $U$ to an item in $V$. For example below, the graph on the left is bipartite, whereas on the graph on the right is not. Provide an algorithm which determines whether or not a graph is bipartite. What is the runtime of your algorithm?

*Hint:* Can you modify an algorithm we already know?

(b) Consider the following implementation of DFS, which contains a crucial error:

create the fringe, which is an empty Stack
push the start vertex onto the fringe and mark it
while the fringe is not empty:
  pop a vertex off the fringe and visit it
  for each neighbor of the vertex:
    if neighbor not marked:
      push neighbor onto the fringe
      mark neighbor

First, identify the bug in this implementation. Then, give an example of a graph where this algorithm may not traverse in DFS order.

*Hint:* When should we be marking vertices?

(c) *Extra:* Provide an algorithm that finds the shortest cycle (in terms of the number of edges used) in a directed graph in $O(EV)$ time and $O(E)$ space, assuming $E > V$. 