EECS3101 notes:: Graph algorithms practice

Jerry Wu

2023-03-28

Exercises on MST

Example 1:: Prove the theorem

"Feel the pressure"-Larry YL Zhang 2023

- Let $G = (V, E)$ to be a connected, undirected, and weighted graph.
- Let $A \subset E \in$ some (partially grown) MST for G.
- Let $(S, V - S)$ be any **partition** of G that represents $A \subset E$. No $\epsilon \in A$ crosses S and $V - S$.
- Let (u, v) be a **min weight** edge crossing S and $V - S$.
- Then (u, v) is a *safe* (minimum weight) edge for A, i.e. (u, v) must belong to some MST.

"Deep down inside, we know that this edge must be in the MST"-Larry YL Zhang 2023

To prove this, we will use a contradiction. This is a greedy algorithm, so we can use the same method as the one we used for the greedy proofs. Take two $\epsilon \in E$. Assume that (u, v) is a minimum weight crossing edge, and (x, y) is also a minimum weight crossing edge. By this, we know that $|(u, v)| \leq |(x, y)|$. Assume that $(x, y) \in SO$ for the sake of contradiction. Adding an extra edge to any tree will create a cycle $\implies$ not tree. So we must delete one of them, i.e. delete (x, y) since it is a larger weight, and replace with (u, v). So now we have that $W(T') \leq W(T)$, which is a contradiction since we assumed that (u, v) does not belong to some MST, and we found one. So (u, v) is indeed a safe edge.

Example 2:: MST on a modified graph

- Take another undirected, connected, and weighted graph with an MST $T \subset E$.
- Add a new edge (u, v) with weight $w(u, v)$ to G such that a new graph $G' = (V, E \cup \{(u, v)\})$ is created. We have now created a cycle.
- How can we efficiently find a new MST T' of G'? Worst case running time must be $O(|V|)$.

Well, we can do this by removing exactly one edge (the maximum edge more often than not) that is **in the cycle**. In order to find the cycle itself, we can perform DFS. However, the runtime for DFS is $|V| + |E| > |V|$. But this doesn't matter since we're running DFS on the MST $G = (V, T \cup \{(u, v)\})$, which contains exactly $|V| - 1$ vertices by definition of tree. So it is indeed, our runtime is $2|V| \in O(|V|)$.

At home practice:: What if an edge is removed from the graph? How will we update the MST to ensure that it stays as a tree?

Example 3:: Mouse catching game (slide 11)

We want to develop a solution to determine whether the mouse player can win the game or not.
Some things we must consider::

- Can this be modelled using a graph? If so, what kind of graph? Well, we can use an undirected graph since players can move in any direction.
- What does each vertex represent? Each vertex represents a cell on the game board.
- What does each edge represent? Each edge represents the path to an adjacent cell on the board that a player can move to.
- How many vertices and edges are there? There are $m \times n$ vertices and $m \times n$ edges.
- What operation do we need to perform on the graph? We want to find the shortest distance to a cell on the edge of the board for the mouse to travel to. There are exactly $2n + 2m$ edge cells. In the worst case, we need to check all edge cells.
  - To make this efficient, we can just do a BFS for the mouse, cat, and dog respectively instead of doing one for the mouse for all the edge cells. We only do 3 BFS instead of $2n + 2m$ BFS iterations.

- But do we really need a graph? No, because we can use pythagorean theorem since it is a grid. We have $O(1)$ runtime in this case.

"After learning all these fancy data structures and algorithms, don't forget the simple stuff!"-Larry YL Zhang 2023

However, the last point assumes that the game board is a free roaming area with no obstacles. If we do introduce obstacles (walls) at random cells, we don't have as much freedom, therefore shortest path can't be obtained by calculating distance alone. This problem would actually require graphs and BFS. So our efforts in the previous example aren't wasted!

Example 4:: Kindergarten management

- Suppose you are a kindergarten teacher who is given a task to arrange n ill behaved children in a straight line, facing forward.
- You are given m statements in the form "kid i hates kid j". i.e. don't put them beside each other!

This is a **topological sort** problem. Each vertex in the graph (queue) represents a kid, and each edge represents the statement "i hates j".

"If they ask you to code it, just implement DFS! It only takes 3 minutes to code!"-Larry YL Zhang 2023