CS325: Analysis of Algorithms, Winter 2024

Practice Assignment 3 – Solutions

Problem 1.

(a) Find a graph that has multiple minimum spanning trees.

(b) Prove that any graph with distinct edge weights has a unique minimum spanning tree.

(c) Find a graph with non-distinct edge weights that has a unique minimum spanning tree (can you generalize (b)?).

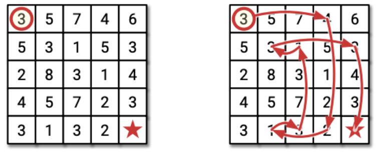
Solution.

(a) Consider a triangle: a graph with vertex set $\{u, v, w\}$ and edge set $\{\{u, v\}, \{u, w\}, \{v, w\}\}$, with all weights equal to one. This graph has three spanning trees with edge sets (i) $\{\{u, w\}, \{v, w\}\}$, (ii) $\{\{u, v\}, \{v, w\}\}$, and (iii) $\{\{u, v\}, \{u, w\}\}$. All these spanning trees have total weight 2, so all of them are minimum spanning trees. More generally, for any graph with equally weighted edges, any spanning tree is a minimum spanning tree. (The number of spanning trees of a graph can be quite large; as large as n^{n-2} for a complete graph with n vertices according to Cayley's formula. Related to this is Kirchhoff theorem, a nice classic result in algebraic graph theory.)

(b) See Lemma 7.1 in the Book.

(c) Any tree has trivially only one minimum spanning tree, even if it has non-distinct edge weights. For a more interesting example, consider a cycle with all its edge weights being 1 except for one edge that has weight 2. The only minimum spanning tree of this graph is obtained by dropping the heaver edge.

Problem 2. A number maze is an $n \times n$ grid of positive integers. A token starts in the upper left corner; your goal is to move the token to the lower-right corner. On each turn, you are allowed to move the token up, down, left, or right; the distance you may move the token is determined by the number on its current square. For example, if the token is on a square labeled 3, then you may move the token three steps up, three steps down, three steps left, or three steps right. However, you are never allowed to move the token off the edge of the board. Describe and analyze an efficient algorithm that either returns the minimum number of moves required to solve a given number maze, or correctly reports that the maze has no solution. For example, given the number maze in the figure below, your algorithm should return the integer 8.



Solution. Construct the *directed* graph $G = (V, E)$ as follows. Let the vertex set $V = \{(i,j) | 1 \leq i, j \leq n\}$; hence there is a vertex for each cell of the maze. For $(i,j), (p,q) \in V$, let $(i,j) \to (p,q) \in E$, if the rules of the maze allow for jumping from cell (i,j) to cell (p,q). Specifically, if one of the following conditions hold.

(1) $p = i$ and $q = j + B_{i,j}$,

(2) $p = i$ and $q = j - B_{i,j}$,

(3) $p = i + B_{i,j}$ and $q = j$ or

(4) $p = i - B_{i,j}$ and $q = j$,

where $B_{i,j}$ is the number written on the board at cell (i,j).

Any jump in the game from any cell (i,j) to any cell (p,q) corresponds to taking an edge of G between the corresponding vertices. Therefore, any sequence of jumps of length ℓ from any cell (i,j) to any cell (p,q) corresponds to a walk in G of length ℓ. In particular, this is true for any sequence of jumps from $(1,1)$ to (n,n). Hence, to find the minimum number of moves to solve the maze, we can instead compute the shortest path length from $(1,1)$ to (n,n) in G. As we have seen in class this can be done using BFS (as G is not weighted).

The number of vertices of G is n^2, one edge per cell of the maze. The number of edges of G is at most its number of vertices times 4, $4n^2$, since each vertex has at most four outgoing edges. The running time of BFS is $O(V + E)$, which in our case is $O(n^2 + 4n^2) = O(n^2)$. Also, we can construct G in $O(n^2)$ time by going trough all vertices and check for possible edges. Therefore, the running time of the algorithm is $O(n^2)$.