HW 8 Solution

CS/ECE 374 B: Algorithms & Models of Computation, Spring 2020 Version: **1.0**

1 Problem 22:

Solution 22:

Let the input sequence of data tuples are

$$(P_{11}, P_{12}, t_1), (P_{21}, P_{22}, t_2)...(P_{m1}, P_{m2}, t_m)$$

where $t_1 \leq t_2 \leq t_3 \leq ... \leq t_m$, and the given observation be (P_x, t_x). That is, the length of the input sequence is m. In this case, we can construct a graph G = (V, E) with vertices and edges defined as follows:

- V = $\{(P_i) \mid 1 \leq i \leq n\}$ that each vertex represents a person in the graph after t_x. Since there are totally n people, we can trivially get that there are $O(n)$ vertices in the graph. And we would need an auxiliary array T to store whether a person has contracted the virus, where T should all be initialized to 0 and T[x] = 1 at time t_x.
- E = $\{v_i \to v_j | v \in V\}$, that is $(P_i) \to (P_j)$ for every pair (P_i, P_j, t_k) for every $x \leq k \leq m$, there exists a pair containing P_i, P_j and $T[i] = 1, T[j] = 0$. In other words, we have an edge in this graph from v_i to v_j if and only if $P[i]$ is infected and $P[j]$ is not. Therefore, we have the following conditions for constructing the edges:
  - if T[i] = 1 and T[j] = 0, we insert an edge in the graph that $v_i \to v_j$, and T[j] = 0.
  - if T[i] = 0 and T[j] = 1, we insert an edge in the graph that $v_j \to v_i$, and T[i] = 0.
  - Else, don't create any edge and continue.

  In this case, we have $O(m)$ edges in the graph and this graph is directed.
- Within this construction, we can set the starting point of this graph at P_x with given P_x, t_x and only consider the tuples after t_k. And all the reachable points from this starting point should be contacted by virus by the end of the algorithm.
- For the termination of this algorithm, as long as we have visited all the reachable points from the starting point, that is the result of the algorithm.
- To solve this problem, we can define the starting point as P_x as given in the context, and we can compute all the reachable points starting from (P_x) by calling ***whatever-first-search*** since both BFS and DFS would satisfy this problem. And the time complexity of this algorithm is $O(V + E) = O(m + n)$. However, since we only have edges when there exists a pair in the tuple sequence. Therefore, the approximate runtime should be bounded in $O(m)$.

2 Problem 23:

Solution 23:

Let the input map T[1...n][1...n] be given. We can construct a graph G = (V, E) with vertices and edges defined as follows:

- V = (x, y, v_0, v_1) that x and y represent the position of the car and v_0 and v_1 represent the velocity at this position. Since we need to only consider the positions within range, (x, y) is a valid position if and only if T[x][y] = 1, with the range $1 \leq x \leq n, 1 \leq y \leq n$. Then we need determine the upper bound of velocity (v_0, v_1).
  - Suppose in the worst case, we increase horizontal velocity every time by 1 at each position. Therefore, we can easily get that the horizontal velocity would be 0, 1, 2... t. Since the position of the car needs to be in the map, the sum of these velocity should be less than equal to n. Thus, $\sum_{v=0}^{t} v = \frac{t(t+1)}{2} \leq n$. In this case, we can get that the approximate upper bound for horizontal velocity is $\sqrt{2n}$. The similar illustration can be applied to vertical velocity too.

  In this case, we would totally have $n \times n \times \sqrt{2n} \times \sqrt{2n} = O(n^3)$ vertices in the graph.

- E = $\{$(x, y, v_0, v_1) $\rightarrow$ (x', y', v_0', v_1')$\}$ where two tuples represent two consecutive positions of the car, where both positions should satisfy the above conditions, and each parameter should satisfy the following requirement:

$$x' = x + v_0 \qquad v_0 - 1 \leq v_0' \leq v_0 + 1$$
$$y' = y + v_1 \qquad v_1 - 1 \leq v_1' \leq v_1 + 1$$

  Therefore, for each vertex (x, y, v_0, v_1), it can have at most 9 neighbors for the next possible move since v_0 and v_1 respectively have three cases. Thus, $|E| \leq 9 \times |V| = O(n^3)$. In this case, we have $O(n^3)$ edges in the graph, which is a directed graph.

- We would also need one more node for the start of the graph. Let's define this start 's', where s has outgoing edges which reach all points in the starting area, i.e., (0, y, 0, 0). In this case, the number of edges coming from s is $O(n)$

- We would also need one more node for the end of the graph. Let's define this end 't', where t has incoming edges from all points in the finishing area i.e., (n, y, v_0, v_1). In this case, the number of edges going into t is $n \times \sqrt{2n} \times \sqrt{2n} = O(n^2)$. Therefore, the extra starting and ending point would not affect the complexity of vertices and edges.

- To solve this problem, we need to find the shortest path from starting point to ending point, which represent the shortest path in bitmap plus two moves, one from starting to starting area, and one from finishing area to ending. We can compute this shortest path by calling ***breadth-first-search***, which return the shortest sequence of moves from start s to end t. This algorithm can be computed within $O(V + E) = O(n^3)$.

3 Problem 24:

Solution 24:

Let the input graph G=(V, E), where |V| = n and |E| = m. We can construct a new graph G'=(V', E') where V' and E' are defined as follows:

- V' = {v × {3, 7, 4} | v ∈ V}. We should leave the vertices unchanged and assign them with value with 3, 7, 4 to check the correctness of the 374 walk, since we need to examine every node that is possible in the 374 walk. Therefore, we have $O(n)$ vertices in this graph.
- E' = $v_1 \to v_2$ where v_1 and v_2 are defined as follows:
  - $(v_1, 4) \to (v_2, 3)$ for every '3' edge $v_1 \to v_2 \in E$
  - $(v_1, 3) \to (v_2, 7)$ for every '7' edge $v_1 \to v_2 \in E$
  - $(v_1, 7) \to (v_2, 4)$ for every '4' edge $v_1 \to v_2 \in E$

  In this way, we can confine all the paths only within 3 followed by 7 followed by 4 followed by 3. And in this graph we have $O(m)$ edges and the graph is directed.
- In our new graph G', we can define the starting point as (v, 7) where v is the starting point of the original graph, that is vertex v given in the context. And to solve this 374 walk problem, we can find all reachable points starting from v in G' as defined above. We can compute this problem by calling ***whatever-first-search*** since both BFS and DFS satisfy this problem and the time complexity of this algorithm is $O(V' + E') = O(m + n)$.