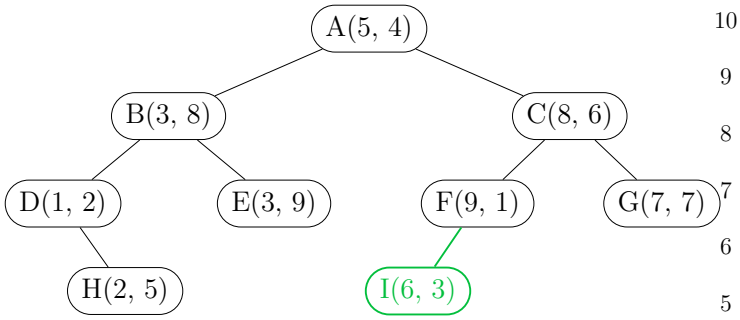
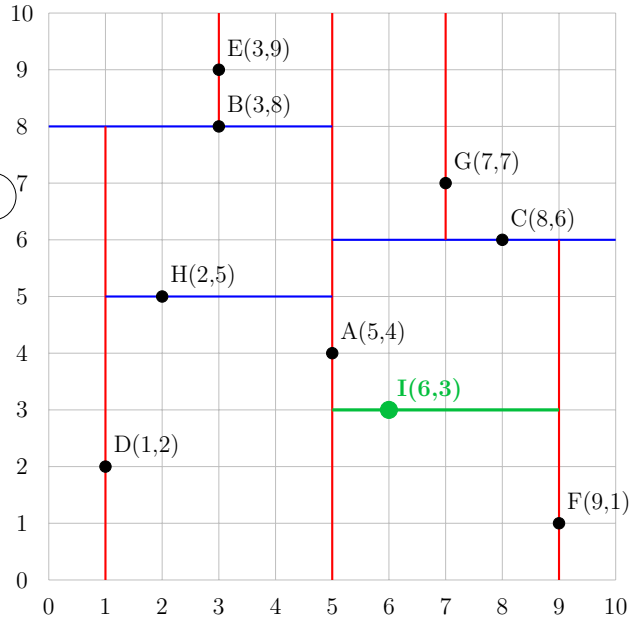
1 K-d Tree

Use the k-d tree and the points in the grid below to answer the following questions.





(a) Insert the point I(6, 3) into the k-d tree above. Then, add that point to the grid and draw the corresponding splitting plane.

(b) The following questions will focus on the pruning process.

i) [Warm up] Using the Euclidean distance, $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, calculate the distance between the following points (the answer can be in the form of square root.)

- T(3, 7) and A(5, 4) $d_{TA} = \sqrt{2^2 + 3^2} = \sqrt{13}$
- T(3, 7) and B(3, 8) $d_{TB} = \sqrt{0^2 + 1^2} = 1$
- T(3, 7) and G(7, 7) $d_{TG} = \sqrt{4^2 + 0^2} = 4$
- T(3, 7) and I(6, 3) $d_{TI} = \sqrt{3^2 + 4^2} = 5$

ii) To find the nearest point to T(3, 7), which side is considered the **good** side (or subspace) splitting by point A(5, 4)?

● Area of $x < 5$ ○ Area of $x \geq 5$ ○ Area of $y < 4$ ○ Area of $y \geq 4$

iii) How about the **bad** side?

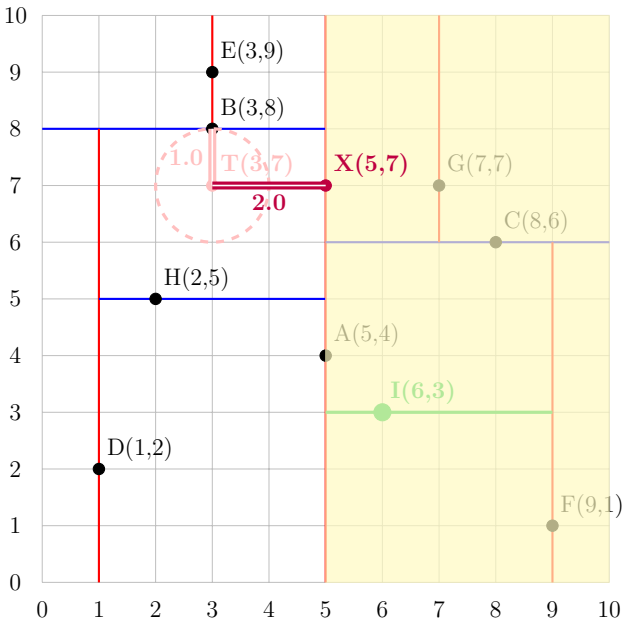
○ Area of $x < 5$ ● Area of $x \geq 5$ ○ Area of $y < 4$ ○ Area of $y \geq 4$

iv) From just looking at the grid above, which known point from the **good** side is the closest point to T(3, 7)? [Note: this process will be done recursively, but now we focus on the recursive call on point A] B(3, 8)

v) Now, we have explored the point A and the **good** side splitting by point A. It's time to decide whether we want to explore the **bad** side splitting by point A (pruning).

- Give an x-y coordinate from the **bad** side (or subspace) such that its distance to T(3, 7) is the smallest among any points from the **bad** size: X(5, 7)
- What is the distance from that point, X, to T(3, 7)? $\sqrt{2^2 + 0^2} = 2$
- By knowing the information from Problem i, iv and v, should we explore the **bad** side splitting by A? ○ Yes ● No ○ Not enough information

Will you want to pick up your worksheet later? Circle one: Yes / No
How confident do you feel with the material this week? Circle one: 1 / 2 / 3 / 4

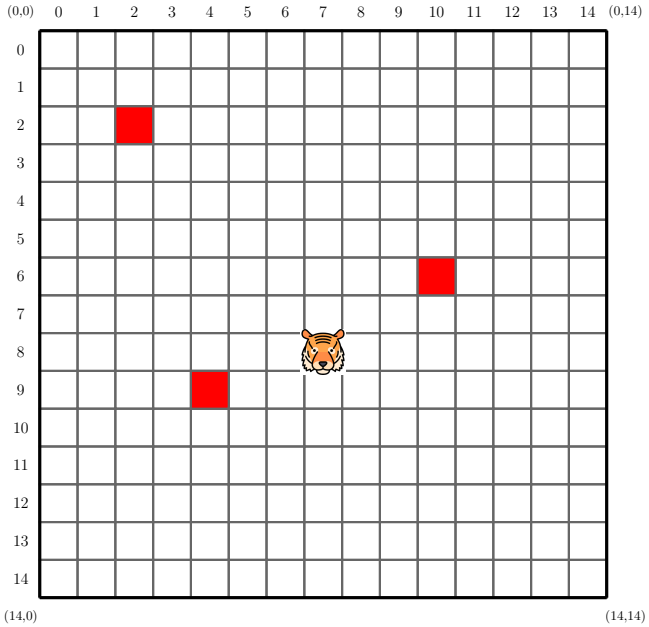


The distance from T to X represents the lower bound for the distance from T to any points in the bad side (yellow area). Hence, the distance from T to any points in the yellow area will be more than or equal to 2.0. After exploring the good side, we know that B is the nearest point so far, with 1.0 distance from the target. Then, we will not explore bad side because the distance will never get to be better than 1.0 which is the best so far.

Will you want to pick up your worksheet later? Circle one: Yes / No
How confident do you feel with the material this week? Circle one: 1 / 2 / 3 / 4

2 Breadth-First Search

On the right is a grid representing a 2-D forest where each red cell represents fire. We want to know whether the tiger can go out of this wildfire forest (out of the border) safely. At each second, the tiger can only move in one of these four directions: Left, Right, Up, Down. Each fire will spread in all four directions at each second. Assume that the forest will only have one tiger and zero or more fire cells.



(a) Give an algorithm to solve this problem using BFS. Explain it in English in a **numbered list of steps**. (Hint: you can change the color of the cell; use `Queue`)

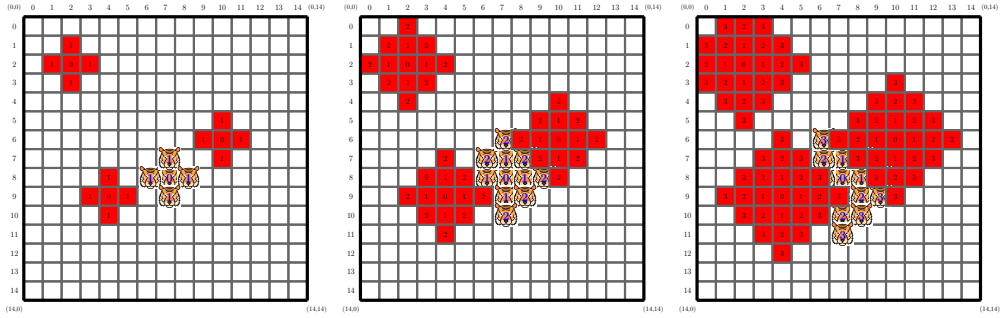
1) put every fire cell into a queue followed by the tiger cell [we are doing multiple-source BFS]
2) pop the cell, `c`, in the front of the queue
3) if `c` represents **fire**:
   for each neighbor next to `c`:
   - if it's still in the bound of the forest AND not yet red, change it to red and put this cell into the queue
4) (else) if `c` represents **tiger**:
   for each neighbor next to `c`:
   - if it's out of bound, **return `true`**
   - else, if it's white, mark it as tiger and put this cell into the queue
5) while the queue isn't empty, repeat at step 2
6) **return `false`** if goes out of while-loop

(b) Give a worst-case runtime of your algorithm in term of W, the width of the grid, and H, the height of the grid.

$\Theta(\underline{WH\quad})$

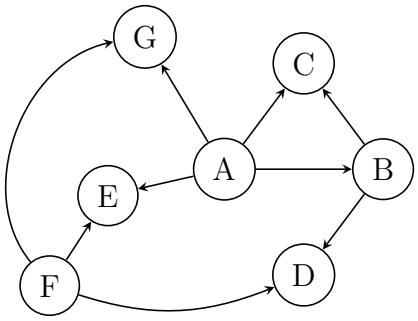
We are doing BFS on the 2-D grid here (with the idea of flood fill). Each cell represents a vertex with at most 4 edges connected itself with its neighbors. Hence, there are WH vertices and at most $4WH$ edges in total. BFS in the graph runs in $\Theta(|V|+|E|)$. Then, this algorithm will run in $\Theta(WH+4WH)$ which is simplified to $\Theta(WH)$

Visualization of first few layers of the flood fill algorithm (zoom in to see the layer order):



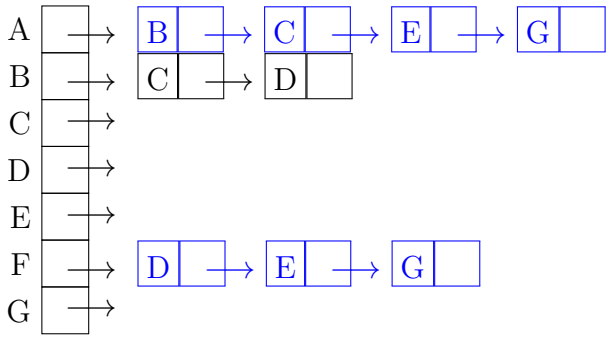
Will you want to pick up your worksheet later? Circle one: Yes / No
How confident do you feel with the material this week? Circle one: 1 / 2 / 3 / 4

3 Graph Implementations and Depth-First Search



(a) Draw the adjacency list that represents the directed graph on the left. (a list for node B is provided for you.)

One possible answer:



(b) Give the order of DFS calls starting from vertex A. Visit neighbors in alphabetical order.

A B C D E G ___