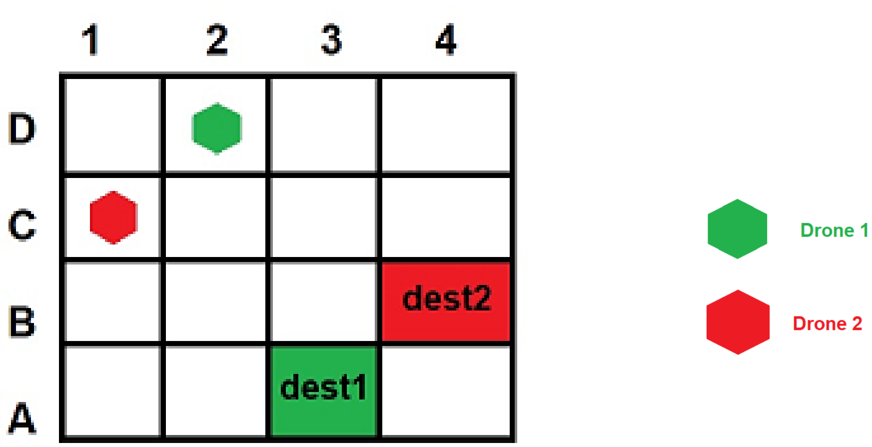
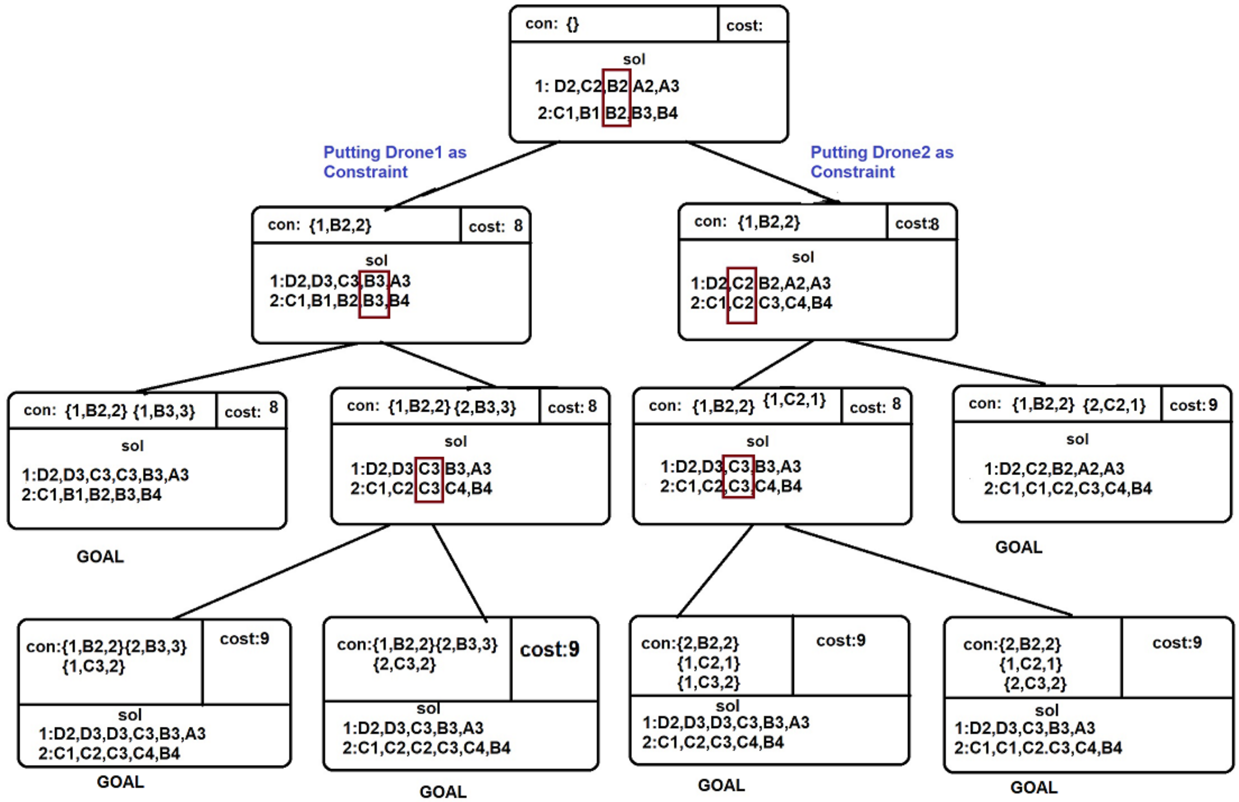
Problem statement

- Assume a situation where multiple drones are flying in 2-D space autonomously. It will create a lot of mess if there is no central software to guide them the path they have to follow to reach their destination.You can assume this 2-D plane to be a M x N grid
- Your task is to design an algorithm which solves the above problem and gives each drone a path which it can follow to reach its destination.It takes input of the number of drones and a list of drones with their starting position, end position, time at which they will start.
- Example Input: [[x11, y11, x12, y12, t1], [x21, y21, x22, y22, t2],.................................[xn1, yn1, xn2, yn2, tn]] • This input tells us that there will be a drone that will appear at point (x11, y11) at time t1 and wants to reach its destination point (x12, y12).
- Similarly, there are certain other drones which will appear at their starting position at their corresponding time, and their target is to reach their target destination.
- Your output should be the path for each drone which can be followed by the drone to reach its destination.
- A path is nothing but just a sequence of positions which are adjacent to each other. You can assume that each drone can move from its position to its adjacent position in 1 second.
- You can assume 8-adjacency i.e all 8 cells around a particular cell are adjacent to it implying that the drone can move in all 8 directions.
- If you wish to go for 4-adjacency i.e assuming that the drone can move only in 4 directions forward, backward, left and right, then this can also be considered.
- The algorithm should be designed in such a way that there occurs no collision between the drones and drones should possibly reach their destination in minimum time possible. You can assume that size of each drone is 1*1 unit and total grid size to be 20*20 units. Apart from all this, you are free to assume some suitable and valid assumptions.

Algorithm Approach





Terminology

- We construct a Tree for each and every possible state of drone names Constraint tree, This is a binary tree and each node N consists of
  - **Constraints:** Each of these constraints belongs to a single drone, The root contains empty set of constraints
  - **Solution:** Set of k paths, for k drones.
  - **Total Cost**:As per our example problem, we can consider total cost as the total time taken by the drone to reach from source to destination with penalty(waiting time) to avoid collision.
- A Node 'N' is considered as goal node when solution is valid(has no conflicts)

Working of algorithm

- Initially root contains an empty set of constraints
- The cost in root node will be '8', as the optimal solution from each drone from its source to destination is <D2,C2,B2,A2,A3> for **d1**, and <C1,B1,B2,B3,B4> for **d2**
- Now while validating the two-drone solution given by the two individual paths, as conflict is found when both the drones arrive at vertex **B2** at time step **2**
- This creates a conflict (d1,d2,B,2), hence the root node is declared as non-goal node and two children are generated in order to resolve the conflict.
- The left child,adds the constraint (1,B2,2) for drone **d1**, while the right child adds the constraint for drone **d2**
- Now the algorithm is performed for the left child to find ans optimal path that also satisfies the new constraint. For this drone **d1** must wait one time step either at **C3**,**D3** or **D2** and the path (D2,D3,C3,C3,B3,A3) is returned for d1.
- The path for d2 (C1,B1,B2,B3,B4) remained unchanged for d2 in the left child.
- The total cost for the left child is increased by 1 and is **'9'** now,because of the imposed penalty.

- Both the children are inserted into our data structure and in the next iteration the left child is chosen for expansion, and the underlying paths are validated.
- Since no conflicts exist, the left child is declared a goal node and its solution is returned as an optimal solution
- We can expand this algorithm for more than k > 2 drones as well, but the implementation and handling of nodes becomes slightly modified

Pseudocode

```
Input: Drone array with source,destination, time of start
Root.constraints = {}
Root.solution = find the individual path
Root.cost = maximum time that might take
Insert Root into OPEN list
While OPEN is not empty do
    X = Best node from OPEN list //Lowest solution cost
    Validate the paths in X until a conflict occurs
    if( X has no conflicts) then
        Return X.solution //X is the goal
    C = first conflict in X

    For each drone d_i in C do
        N = new node
        N.constraints = X.constraints + (d_i,v,t)
        N.solution = X.solution
        Update N.solution
        N.cost = update the cost with the wait penalty
        if(N.cost < infinity) //A solution was found then
            Insert N to OPEN
```