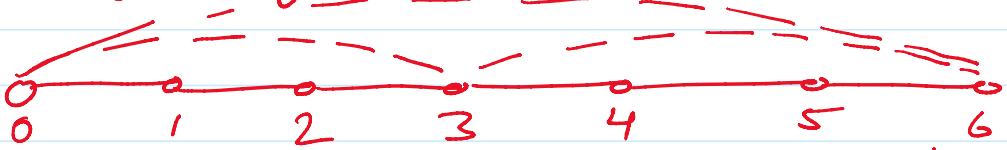
Thursday, September 19, 2019 4:58 PM

Q1 Girth: Do a BFS starting from vertex $v \in V$. If we find an edge $e=(a,b)$ which forms a cycle then we estimate length of cycle as level(a) + level(b); note that this is an upper bound. The BFS is done from all vertices in the graph & the smallest cycle found gives the girth. Note that when we do BFS from a vertex which is on the smallest cycle then the estimate of the girth is the correct value. The running time of this algorithm is $O(mn)$

Incorrect answers:

1) Doing a DFS & considering cycles formed by one back edge & some tree edges is incorrect. This is because the shortest cycle might only include back edges as this example shows



The back edges are dashed. The girth is 3 but if we consider only cycles with 1 back edge, the smallest such cycle has length 4.

Q2) Orienting edges of a 2-edge connected graph to form a strongly connected graph:

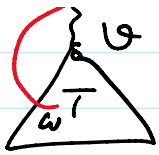
It is true that the edges of a 2-edge connected graph can always be oriented to form a strongly connected graph. Do a DFS from an arbitrary vertex s, orient tree edges away from s & all back edges towards s; let G' be the graph obtained.

We now prove that the directed graph so formed is strongly connected.

a) Note that there is a path from s to every vertex in the graph using only tree edges

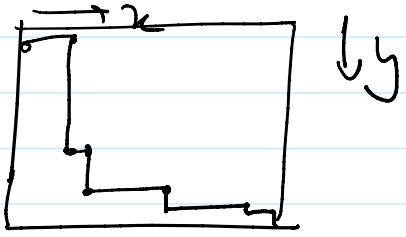
b) Consider a vertex v & let T be the subtree rooted at v. Since G is 2-edge connected there is a back edge from T to an ancestor of v, say u. Since arrival[u] < arrival[v] we can go from v to u in the directed graph by following the tree edges from v to u & the

We can go from v to u in the directed graph by following the tree edges from v to w & the back edge (w,u). Hence from every vertex v in the graph we can reach a vertex of lower arrival time. Repeating this implies we can reach s & hence there is a path from any vertex v to s. This in turn implies G is strongly connected



Q3) Let A be an n×n matrix s.t. $A[i,j] = 1$ iff there is a coin in this cell.

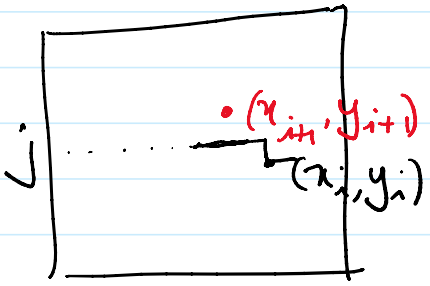
The robot goes to the first column which has a coin & collects all coins in that column. At the last coin it moves right & goes to the next column to the right which has coins it can collect. If goes to that column & repeats till it reaches the bottom right corner. If there are coins remaining we start with another robot & follow the same procedure



Let k be the no. of robots used. We identify cells (x_i, y_i) $1 \le i \le k$ s.t. $A(x_i, y_i) = 1$ & the coin in this cell was picked by robot i

a) (x_k, y_k) is the cell at which Robot k makes the first left turn.

b) (x_i, y_i) is the first cell in rows $> y_{i+1}$ at which robot i makes a left turn.



Note $y_i > y_{i+1}$ $1 \le i \le k-1$.

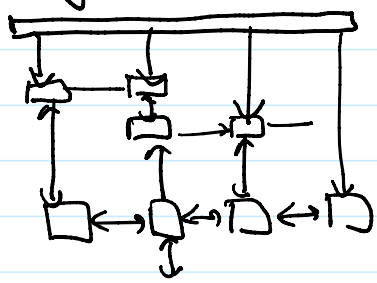
Claim: $x_i < x_{i+1}$ $1 \le i \le k-1$

Proof: Suppose $x_i \ge x_{i+1}$. When Robot i was crossing column x_{i+1} it was on a row below y_{i+1} or else it would have picked the coin in (x_{i+1}, y_{i+1}); suppose this was row j. So when robot i turned into row j it made a left turn & hence (x_i, y_i) was not the first left turn after row y_{i+1}

Thus we have $x_1 < x_2 < x_3 \ldots < x_k$ and $y_1 > y_2 > y_3 \ldots > y_k$. A robot which visits one of these cells cannot visit the other & hence at least k robots are

$y_1 > y_2 > y_3 \cdots > y_k$. A robot which visits one of these cells cannot visit the other & hence at least k robots are required in any solution.

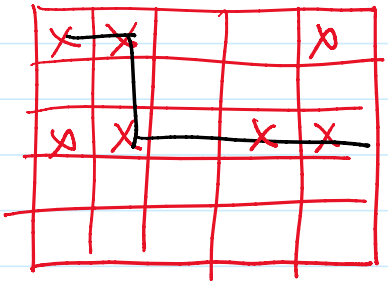
Implementation: We implement Δ using the sparse matrix representation in which every row/column is a doubly linked list of cells in that row/column containing a coin. An array has links to the head nodes of the linked lists for each column. We go to the first element of the array which is not null & follow list down till the next pointer is null. Continue by taking the row.next pointer & again follow the column.next pointers till they become null. All nodes encountered are deleted & this can be done in constant time for each node since the lists are doubly linked. Process is repeated for the next robot.



Total time needed is just the no. of nodes which is the no. of coins. which is at most n^2. To create this representation from the matrix representation of coin positions we would need $O(n^2)$ time.

Wrong solutions:

1) going to the nearest coin would lead to the first robot taking the black path. This would mean 3 robots while the minimum is 2.



2) finding a path which covers most coins would again take Δ along the black path first. This would mean 3 robots while optimum is 2.