Homework #4

1. a) Proof by contradiction

Assume any pair of vertices in a tree are connected by more than one path

Let x & y be the vertices chosen to be connected.

Let the following be the vertices to consider (in our tree) V : $v_1, v_2, v_3, v_4, \ldots, v_{n-1}, v_n$

Then assume a possible path from x to y is:

$\{x, v_1\}, \{v_1, v_2\}, \{v_2, v_3\} \ldots \{v_{n-1}, v_n\}, \{v_n, y\}$

To create another possible path, say there is a vertex v_j that is adjacent to v_x & v_y, where v_x & $v_y \in \{v_1, v_2, v_3 \ldots v_n\}$

Now we can go from x to y with path 1 & go back to x from y from path 2.

This creates a cycle in our tree. But by definition of a tree, this is not an appropriate characteristic (a cycle).

Therefore this is a contradiction.

So the original statement is true.

b) Direct proof: since this graph is a tree, any pair of vertices are connected by 1 path (proven in 1a). Choose our pair to be $x, y \in V$. Assume these two vertices are current not connected with eachother. So by definition of a tree, there's current a path to it, passing the follow vertices $v_1, v_2, v_3 \ldots v_n \in V$. Then if we add an additional edge x, y, we have created a cycle because there is a way to go from x to y & back to x without retracing the path. Hence creating a circle.

c) Proof by Contrapositive

If for every pair of vertices in the graph, they are connected by more than one simple path, then the graph must not be a tree.

This first part of the statement means that for $x, y \in V$, there is a path $\{x, v_1\}, \{v_1, v_2\}, \{v_2, v_3\} \ldots \{v_n, y\} \in E$ & $\{v_1 \ldots v_n\} \subseteq V$. And there is at least another path to get from x to y. Say this path is $\{x, v_1'\}, \{v_1', v_2'\} \{v_2', v_3'\} \ldots \{v_n', y\} \subseteq E$ & $\{v_1' \ldots v_n'\} \subseteq V$.

Then we can go from x to y passing through different vertices than we would going backwards. This allows us to go in a loop/cycle & as a result, by definition of a tree, this graph is not a tree b/c it has a cycle.

Because this is the contrapositive to our claim, the original claim is true. ∎

d) Proof by Contradiction

Say a graph has no simple cycle and has the property of adding certain Single edges will not create a simple cycle, then the graph is a tree.

Then there are vertices $x, y \in V$ where $\{x, y\} \notin E$. From our assumption, assume there is no path connecting x & y. in this graph. So when we add the single edge $\{x, y\}$, it creates no cycle because currently there is only one path (newly created) going from x to y.

However this tells us the original graph was disconnected, and by the definition of a tree, the graph was not a tree. So claim is false.

The claim proven is a contradiction to the original claim, so the original claim is true.

2. a) • First, assume $\forall v \in G$, indegree(v) = outdegree(v), and our G is connected.

• Then understand the definition of stuck:

when we reach a vertex, there's no outgoing edge that has yet to be traversed

• Note that all the vertices we passed on our arbitrary walk have the same # of in & out degrees.

• Realize at the start of our walk, at vertex s, we will be using/traversing the outgoing edge. And since its # of in & out degrees are the same, our vertex s will run out of outgoing edges first before its incoming edge.

• This means any other vertices on our walk will be using its incoming edge first since its only possible to get from vertex$_A$ to vertex$_B$ from an incoming edge to vertex$_B$. This means for all vertices (not s) will run out of incoming edge(s) before outgoing edg(s).

• By our definition & reasoning, s has to be the starting point & ending point b/c all other vertices, if we can get there, will at least have an outgoing edge b/c of our previous statement. Whereas s will be the opposite since we left & used its outgoing edge first. Therefore when we get there, we will be stuck by definition.

• And this is a tour b/c we did not allow to traversed over previous edges — hence why we "run out". □

a) i. • Assume it is the same: directed & connected. All vertices in the graph: outgoing(v) = incoming(v)
• In our previous question, our tour, we started at s & ended at s.
• Know that getting stuck means S has no more outgoing edges & since we proved/explained that outgoing edges will run out after incoming edges, by the time we get stuck, we have used up all of S. Therefore we do not need to consider this vertex.
• Now we will prove that all the vertices have equal number of outgoing & incoming untraversed edges.
• Going back to our original proof, all vertices that were not part of the tour will still have equal outgoing & incoming edges b/c we never traversed over any of the edges incident to it since if we went on the incoming edge, it means we made that vertex part of the tour. And you need to be part of the tour in order to continue & go on the outgoing path.
• Then notice our tour ends at s, therefore whatever vertices we are at we must leave on an outgoing edge & to get there, we must have entered through an incoming edge. Since both happens consecutively, the ratio between incoming edge & outgoing edge will stay the same. Therefore incoming edge(v) = outgoing edge(v), where $\forall v \in G$, after the cycle still. □

ii. • Remember our assumption is that the graph is connected and that after our tour, we will have some remaining traverse edges
• Since the graph is connected, we can assume that there is a vertex, v', not part of the Tour in the graph. And for this graph to fall correctly under our assumption, there has to be a vertex with an outgoing to v' (and incoming to v') for the graph to be connected.
• Therefore there is a vertex in our tour with an untraversed edge outgoing. □

2b) Assume • FindTour(G, s) does as described →

• Splice(T ... T_k)

• SubG(G, s, V_x) will take in a graph, a vertex, and a list of vertices w/o any traversed edges & will return us a best-sub graph G′ from G that includes s but no $v \in V_x$.

Function Euler(G, s) :

let V_x be vertices w/o any untraversed edges adjacent to it, assume it automatically updates to findTour

T = FindTour(G, s)

while(V_x still has vertices inside)

G′ = SubG(G, V_x.next, V_x)

T′ = FindTour(G′, V_x.next)

T = Splice(T, T′)

ouput T

End Euler

2c) Proof by Induction on n ; n is the number of vertices

Base Case : n = 1, works!

Hypothesis : for all n ≤ k, assume we can find a tour that starts at at s & ends at s that uses all the edges in the graph without traversing edges already traversed – which makes it a Euler Tour. The graph for this is directed, connected & Outgoing(v) = ingoing(v)

Step: Then for the graph with vertices k+1, we have a graph with k vertices connected with an additional vertex, V_n & k-vertices.

So assume V_n is not connected to the graph

For V_n to be connected to the graph, there must be vertex, v′ ∈ Euler Tour from k-vertices (we know there exists from our hypothesis) with an outgoing edge towards V_n & an incoming edge towards v′ so our graph is correct.

2b Cont.) So when we remove outgoing & incoming edges connecting v' & v_n together, we have a graph with k-vertices. From our hypothesis we can find a Euler tour for a graph w/ k-vertices.

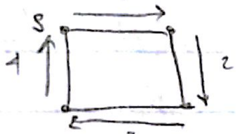
We connect v_n back to the graph with v', we know that there is a possible Euler tour going from v_n & v'; Since the number of vertices here is less than k.

Now we can connect the original Euler tour with the v_n & v' Euler Tour starting at v'. So during our tour, we will stop at v' & take the second tour, which will lead us back to our v' & continue the original tour.

Hence we created a Euler Tour with the graph G w/ k+1 vertices. ∎

3. Proof by induction on n-dimension.

Base case : n = 2. 2 Dimensions, meaning 2^n vertices & has a Hamilton tour (using 4 edges (2^n) that has not been traversed):



* notice all 4 vertices are visited & we took 4 untraversed edges to get there.

Hypothesis : There is a Hamilton Tour for n = k dimensions. This means that all of its vertices (except for the beginning vertex s) is passed/visited exactly once from untraversed edges.

Note that this means within the Tour, there is a Hamilton path except it just doesn't end at s (our beginning vertex), since the path has the same attributes except for ending at s. You can also find one at any ver

Step : let n = k+1 dimensions. Notice the number of vertices = 2^{k+1} from the equation we've established. Expand it to $2(2^k)$. We now see that a hypercube of k+1 dimensions is made of subcubes of k dimension.

Now, with this knowledge, we pick a starting position s for our k+1 dimension cube. Let this vertex s be in our subcube A (there are 2 subcubes). Note in our hypothesis, we said we can find a Hamilton tour in k-dimension & therefore a Hamilton Path. Using this, we will start the path at s & stop when all vertices have been passed once.

3continue) we will call ending vertex v_f. By definition, we have covered $2^k - 1$ edges (untraversed), and all 2^k vertices once. Now we will move onto the other subcube, subcube B. Since this is a hypercube, Subcube A & Subcube B are connected by an edge – you can move from one to another by an edge. We cross this edge to get to vertex s', our starting point in subcube B. This adds one edge to our current tally.

Now on subcube B, at s', we will use our hypothesis (since its 2^k – dimension) to state that there is a path ending at the vertex right before s'. This will take $2^k - 1$ edges (untraversed) to our tally, since we know we are on a completely different subcube. At our end point, it will be directly mirrored (and connected) to our starting position b/c we know k-dimension path will end right before our starting position, which is directly below s. Again s and the vertex mirrored it (where we stopped for subcube B is connected by an edge (by definition of a subcube). We will get back to s (b/c its a tour) by taking the connecting edge right before s' & s. This adds one to our number of edge taking & we've successfully done a Hamilton Tour for 2^{k+1} – dimension cube because:

1. All vertices were past through the 2^k – dimension subcube path
2. Took us to traverse $(2^k - 1) + 1 + (2^k - 1) + 1 = 2(2^k) = 2^{k+1}$ edges.

4. Induction on n, $n :=$ total number of vertices $\in G$.

Base Case : $n=2$ will have a hamilton path representing the tournament where on person will beat the next person.

Two cases : 1) $a \to b$ "a beats b" } hamilton b/c both vertices a&b
2) $b \to a$ "b beats a" are passed & $n-1$ edges were traversed.

Induction Hypothesis : $n=k$, we will find a Hamilton path for this graph such that one person will beat the next (current vertex will have an outgoing ~~vert~~ edge to the next vertex); we find a Hamilton path always starting at the vertex w/ most outgoing edge.

Induction Step : for $n=k+1$ vertices, remove the vertex with the most outgoing edges. By our hypothesis, we can find a Hamilton path for k+1 edges.

Now, if the removed vertex, V_x, ~~has an~~ beats (has an outgoing edge) the current "winning" vertex, V_w, (the vertex we start our path), we connect V_x to V_w & our new path starts at V_x & traverse the same path as when V_w was the head. Otherwise it doesn't beat ~~Vw~~ V_w & we'd remove V_w and put V_x back in & repeat the algorithm until we find a V_w' such that V_w beats (has an outgoing edge). □

Scanned by CamScanner

5. a) 1001 → 0001 → 0000 → 0100

b) The Hamming distance for starting x & ending y is however many bit positions they differ b/c notice we can only travel to 1 bit differences away from our current position. Therefore we'd want to travel one bit at a time until the bits reach/is our destination bits. To do this, we shift each bit we need to change one at a time & therefore will take H(x, y) "changes".

c) shortest length: 2

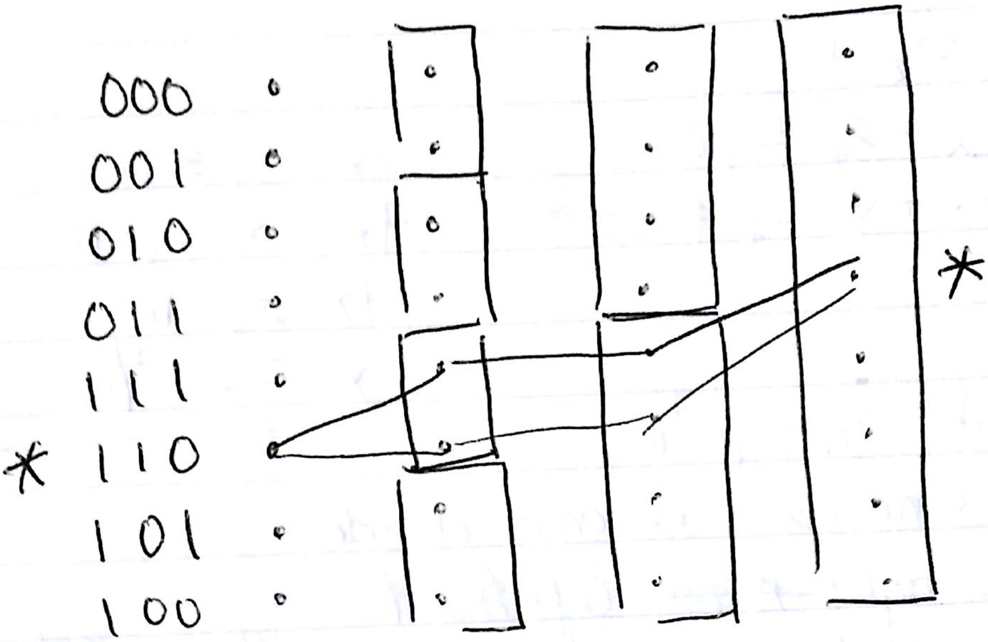
path: 110 → 010 → 011, 110 → 111 → 011

I see that were's given an array of options/routes to take & it'd still get us to our destination

d) To go from vertex x & y existing on a hypercube, we shift, starting from the first bit that is not the same on the left-hand most side. Change it to the one that is the same as the destination. Essentially we are moving onto the same space/plan/universe as we change the bits. Continue to do so until until the changed/new bits look the same as the destination. Notice that we only changed the ones that were different & everytime we change a bit, we "moved" onto that vertex, so we record this to put into our list of vertices. And since we make 1 shift at a time, & we shift for all bits, n, different, our path took n-edges to traverse.

5e)

~~000~~
~~001~~
~~010~~
~~00~~



6) a) $8x \equiv_{15} 1$ then $x = 2, 4, 13, 17, \ldots$

b) $6x + 14 \equiv_{23} 11$ then $x = 11, 34, 57, \ldots$

$6x \equiv_{23} -3$

$6x \equiv_{23} 20$

c) $5x + 13 \equiv_{20} 4$

$5x \equiv_{20} -9$

$5x \equiv_{20} 11$

no solution b/c w/ 5, there can only be ~~≡~~ 0 b/c 20/5 has no remainders - has factors with ~~4~~ 11

d) $5x + 14 \equiv_{20} 4$ then $x = 2, 6, 10, 14 \ldots$

$5x \equiv_{20} -10$

$5x \equiv_{20} 10$

e) $2x + 3y \equiv_7 0$

$3x + y \equiv_7 4$ $\longrightarrow$

$2x + 3y \equiv_7 0$

$-9x - 3y \equiv_7 -12$

$-7x \equiv_7 -12$

$x \equiv_7 +7/12$

$2(7/12) + 3y \equiv_7 0$

$7/6 + 3y \equiv_7 0$

$3y \equiv_7 -7/6$

notice it $3y \equiv_7 -7/6$,

there's no way we can divide the ~~right hand~~ left hand side by 7 such that we get $-7/6$ or $35/6$ as a remainder. Therefore no soln.

Scanned by CamScanner