1. (20 points) Consider a set of intervals $I_1, I_2, \cdots, I_n$:

   (a) Design a linear time algorithm (assume that intervals are sorted in any manner you wish) that assigns the intervals to the minimum number of processors.

   (b) Prove the correctness of your algorithm.

No statement indicating we need processing time to be efficient. → I assume that we can process the intervals in as much time as needed to minimize number of processors, which is 0 or 1 (0 if set of intervals is empty)

Algorithm to assign arbitrary sorted order of intervals to 0 or 1 processor, which is the minimum given constraints:

- for each interval I_i, assign to a processor at next available time, which is after previous interval I_j finishes at $f(I_j)$.
- once all intervals are assigned, terminate algorithm. □

b) Proof:

Base case: 0 intervals → 0 processors ✓
1 intervals → 1 processors ✓

Induction Hypothesis:

assume we can assign $n = k$ intervals $I_1 \ldots I_k$ to 1 processor.

$n = k + 1$

Algorithm assigns $I_1 \ldots I_k$ to 1 processor.
For interval I_{k+1}, algorithm would have to assign after I_k finishes at $f(I_k)$.
If an interval is unassigned, algorithm will assign it.

17/20

Name(last, first): Midterm 10/30/2018

2. (20 points) (a) Design an efficient algorithm that outputs the vertices of a DAG (Directed Acylic Graph), such that if there is an edge (x, y) then x is output before y.

(b) Analyze the run time of your algorithm.

a) A property of a DAG is that it can be topologically sorted. An algorithm that outputs vertices in topological order, which ensures that a vertice x that has an out-degree to y will output before y:

- Start by examining each node in the DAG and checking its in-degree value and out-degrees to other vertices.
- while the subset of nonsource nodes V is not empty:
  - for all nodes with in-degree 0, add nodes to subset of Source nodes S
  - delete nodes added to S from V
  - Now that nodes have been deleted, adjust in-degrees of other nodes
  - Iterate until all nodes are deleted
- output nodes in the order they were added to S

Proof: assume there is a pair of nods (a,b) where b output before a.

- The graph is a DAG so there is no cycle
- If edge (a,b) exists, a has out-degree to b. $a \rightarrow b$
- Running the algorithm, suppose we reach node b that now has in-degree 0. In order for b to reach 0, b cannot have any out-degrees to b.
- One out-degree to b is a. If b in-degree is 0, a must have already been deleted, added to S.
- A output first → this a contradiction, the algorithm in correct.

b) examine each node in DAG : O(V) ✓ subsets of nodes are size V → O(V) to output in order

Iterate through all vertices once, through all edges at least once : O(V + E)

Total O(V + E)

delete edges ? O(e) 7/10

must have even # of vtxes w/ odd degree.

3. (20 points) An undirected graph is said to have property X if you can start from a vertex, traverse all edges of the graph exactly once, without removing your pen from the paper.
   (a) Classify the graphs that have property X?
   (b) Design an efficient algorithm for generating a traversal of a graph that has property X.

I Assume you must reach each vtxes in the graph exactly once by traversing each edge exactly once. I also assume cycles are allowed (so you may reach the starting vtxes 2 times.

a) Graphs with this property must have exactly 2 or 0 vtxes with odd degrees. ✓

b) Given graph w/ above property:

O(V) → Check all vertices for total degree value in completed $G=(V,E)$.

If there are 2 vtxes with odd degrees:
start on one of the odd vertices.

If there are 0 vtxes with odd degrees:
start on arbitrary vtxes.

For traversal, initialize all current degrees to 0.
While there are edges that have not been traversed:
traverse from vtxes x to vtxes y to form (x,y) if incrementing the current degree of the node will not complete node y (where y's current degree = y's total degree in the completed graph $G=(V,E)$)
once you form edge (x,y) increment current degree of x & y: +1 degree

-6

Once the only edge left to traverse will complete y (current = total degrees), traverse the edge.
Algorithm terminates.

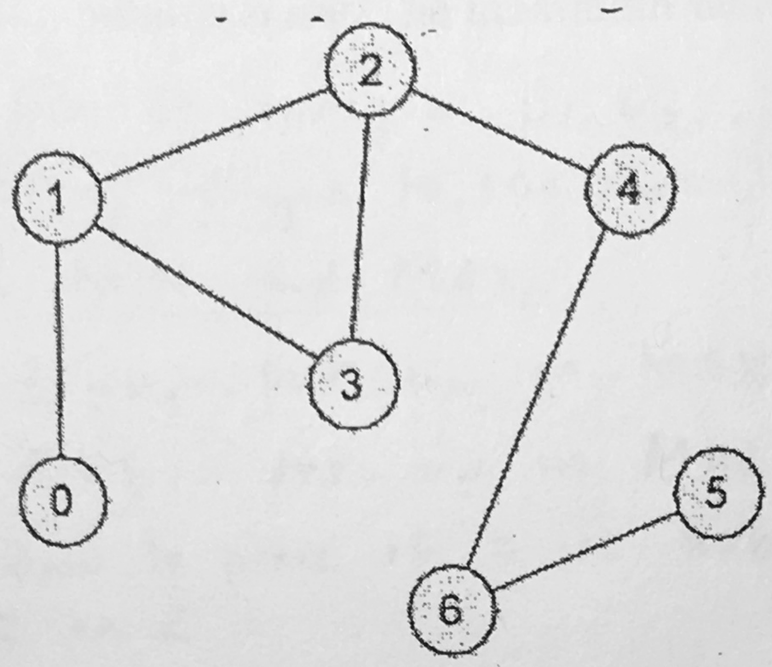
Runtime $O(V+E)$: traverse all vtxes and edges at least once.

does having edge complete a node

10

4. (10 points) Consider an unweighted graph G shown below:

   (a) Starting from vertex 1, show every step of BFS along with the corresponding FIFO next to it.



- start at vrtex 1 {1}

*once a node is pushed to qeue, it is marked as explored and will not be examed again as an adjacent vertice.

- pop 1, examine whics adjacent to 1 : {2, 3, 0}
- pop 2, exame whos adjacent to 2: {3, 0, 4}
- pop 3, exame whos adjacent to 3: {0, 4}
- pop 0, exame whos adjacent to 0: {4}
- pop 4, exame whos adjacent to 4: {6}
- pop 6, exme whos adjacent to 6 : {5}
- pop 5, exme whos adjacent to 5 : {}
- Qeue is empty, all nods explored.

5. (20 points) Consider an unsorted list of integers. You can find the minimum number in the list with $n-1$ comparisons. Similarly, you can find the maximum with $n-1$ comparisons. So you can find both the minimum and the maximum with about $2n-3$ comparisons. Design an algorithm that finds both the minimum and the maximum using about $\frac{3n}{2}$ comparisons.

Given unsorted list of integers $i_1, i_2, \ldots i_n$:

Pick 2 arbitrary integers to compare, i_a & i_b:

2 subsets MIN and MAX:

$\frac{1}{2}n$ compare to sort into 2 subsets.

- if $i_a \geq i_b$, put i_a in MAX and i_b in MIN
- $i_a < i_b$, put i_a in MIN and i_b in MAX.
- * what to place if $=$ is arbitrary doesn't matter if $\geq$ or $<$

Keep comparing until there are 0 or 1 uncompared integers.

In each subset MIN and MAX:

compare 2 arbitrary integers in he subset i_x and i_y:

FOR MIN:

if $i_x \leq i_y$, i_x temporary minimum.
else i_y temporary minimum.

FOR MAX:

if $i_x \geq i_y$, i_x temporary maximum.
else i_y temporary maximum

} each comparison between integers made only once for that pair of integers.

each subset has $\sim\frac{1}{2}n$ integers, so at most you make $\frac{1}{2}n-1$ comparisons to find he minimum of MIN and maximum of MAX.

↳ total $n-2$ for both subsets.

If there is 1 uncompared integer not in either subset (if there is odd number unsorted integers)

2 compares { compare minimum of MIN to that number
compare maximum of MAX to that number

change maximum or minimum value if necessary.

The maximum and minimum you end up with are for he given list. □

$\frac{1}{2}n + n - 2 + 2 = \frac{3n}{2}$ comparisons

6. (10 points) Give an algorithm to color a graph with 2 colors (assuming it is 2-colorable). A proof of correctness is not necessary.

I assume. 2-colorable = bipartite

start at an arbitrary node of the graph.

assign starting node with 1 color A.

BFS search outward.

- If the previous node (the node popped from Queue) is color A, assign adjacent node being explored color B.
- vice versa if previous is B, assign new A.

The colors of nods alternate by level of BFS searching outward from staring node.

Once BFS explores all nodes, terminate algorithm.

Run time $O(V+E)$, explores all vertices and edges. ✓