CSCI 570 EXAM II
TRUE/FALSE
REVIEW

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE] Do not affect capacity of min cut, but may affect choice of min cut
In the Ford-Fulkerson algorithm, choice of augmenting paths can affect the min cut.

[TRUE/FALSE]
A dynamic programming algorithm with n^2 unique subproblems, could have a time complexity that is better than $O(n^2)$.

[TRUE/FALSE]
A dynamic programming algorithm with n^2 unique subproblems, could have a memory requirement that is better than $O(n^2)$. LCS can optimal to O(n) space complexity

[TRUE/FALSE]
A dynamic programming solution when implemented using iteration will always run faster compared with the same dynamic programming solution implemented using recursion and memoization.

[TRUE/FALSE] Pseudopolynomial
It is possible for a dynamic programming algorithm to have exponential running time.

[TRUE/FALSE]
In the final residual graph constructed during the execution of the Ford-Fulkerson Algorithm, there's no path from source to sink.

[TRUE/FALSE]
Bellman-Ford algorithm is guaranteed to work on directed graphs that don't contain any negative weight cycles.

[TRUE/FALSE] Negative cycle
Bellman-Ford algorithm is not guaranteed to work on all undirected graphs.

[TRUE/FALSE]
If we replace each directed edge in a flow network (except for edges incident on *s* or *t*) with two directed edges in opposite directions with the same capacity and connecting the same vertices, then the value of the maximum flow remains unchanged. Directed graph -> undirected graph, cannot always keep the same max flow.

[TRUE/FALSE]
Let (S,V −S) be a minimum (s,t)-cut in the network flow graph G. Let (u, v) be an edge that crosses the cut in the forward direction, i.e., u □ S and v □ V − S. Then increasing the capacity of the edge (u, v) necessarily increases the maximum flow of G.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[/FALSE] The value of the max flow is not enough and we need the complete flow
Given the value of max flow, we can find a min-cut in linear time.

[/FALSE]
The Ford-Fulkerson algorithm can compute the maximum flow in polynomial time.

[/FALSE]
A network with unique maximum flow has a unique min-cut.

[TRUE/]
If all of the edge capacities in a graph are an integer multiple of 3, then the value of the maximum flow will be a multiple of 3.

[/FALSE]
The Floyd-Warshall algorithm always fails to find the shortest path between two nodes in a graph with a negative cycle. **Only affects two nodes with negative cycles**

[/FALSE] 0/1 knapsack does not have a polynomial time dynamic programming solution—not even a weakly polynomial solution. 0/1 knapsack has a pseudopolynomial time dynamic programming solution which is basically an exponential time solution (with respect to its input size)
0/1 knapsack problem can be solved using dynamic programming in polynomial time, but not **strongly** polynomial time. **Should be exponential / psudopolynomial time**

[/FALSE]
If a dynamic programming algorithm has n subproblems, then its running time complexity is $O(n)$.

[/FALSE]
The Travelling Salesman problem can be solved using dynamic programming in polynomial time **Could be exponential time**

[/FALSE] **Only consider two constrains**
If flow in a network has a cycle, this flow is not a valid flow.

[FALSE/] We can't do this for *undirected* graphs
We can use the Bellman-Ford algorithm for undirected graph with negative edge weights. **will have negative cycle**

[TRUE/]
The number of iterations it takes Bellman-Ford to converge can vary depending on the order of nodes updated within an iteration.

[TRUE/]
Maximum value of an s-t flow could be less than the capacity of a given s-t cut in a flow network.

[/FALSE]
Ford-Fulkerson algorithm has pseudo-polynomial time complexity, but the scaled version of Ford-Fulkerson achieves polynomial time complexity by making better choices when selecting s-t paths.

[/FALSE]
In a circulation network with no demands, no supplies, and no lower bounds on flow, the only feasible circulation is one where flow over every edge is exactly equal to zero.

[/FALSE]
On a connected, directed graph with only positive edge weights, Bellman-Ford runs asymptotically as fast as Dijkstra.

[TRUE/]
In the worst case, merge sort runs in O(n^2) time

[/FALSE]
Ford–Fulkerson algorithm is guaranteed to find max flow only if all edge capacities are positive integers. **Consider 0**

[/FALSE]
In a flow network with a unique min cut, if we increase the capacity of the min cut by k, then the value of max flow will go up by k.

[TRUE/]
It is possible that in a divide and conquer solution, the divide step takes longer than the combine step.

2) 12 pts

Solve the following recurrences by giving tight θ-notation bounds. You do not need to justify your answers, but any justification that you provide will help when assigning partial credit.

(a) $T(n) = 3T(n/5) + \lg^2 n$

Solution: By Case 1 of the Master Method, we have T(n) = ( ).

(b) $T(n) = 2T(n/3) + n \lg n$

Solution: By Case 3 of the Master Method, we have T(n) = (n lg n).

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

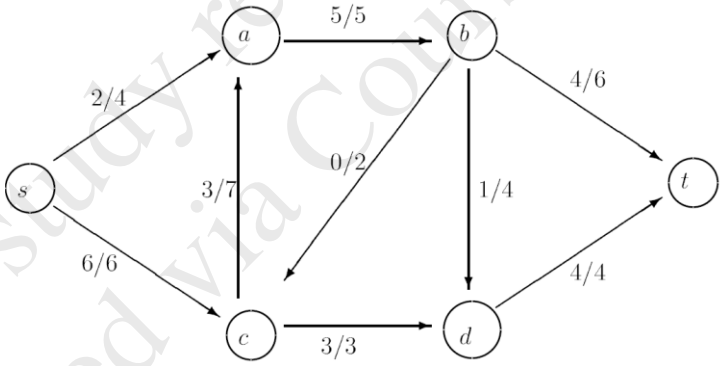
[TRUE]
The number of iterations it takes Bellman-Ford to converge can vary depending on the order of nodes updated within an iteration.

[FALSE]
In a flow network, if the capacity of every edge is odd, then there is a maximum flow in which the flow on each edge is odd.

[TRUE]
Maximum value of an s-t flow could be less than the capacity of a given s-t cut in a flow network.

[FALSE]
Any Dynamic Programming algorithm with n^2 unique sub-problems will run in $O(n^2)$ time.

[TRUE]
The following flow is a maximal flow.
Note: The notation *a*/*b* describes *a* units of flow on an edge of capacity *b*.



[FALSE]

In a circulation network, there is a feasible circulation with demands $\{d_v\}$, if $\sum_v d_v = 0$ Reverse the condition is not correct

[FALSE]

An optimal solution to a 0/1 knapsack problem will always contain the object i with the greatest value-to-cost ratio V_i/C_i.

[TRUE]

The dynamic programming solution presented in class for the 0/1 knapsack problem is not an efficient solution.

[FALSE]

For any edge e that is part of the minimum cut in G, if we increase the capacity of that edge by any integer k>1, then that edge will no longer be part of the minimum cut.

[TRUE]

Ford-Fulkerson Algorithm cannot solve the max-flow problem in a flow network in polynomial time, however, there are other algorithms that can solve this problem in polynomial time. 埃德蒙德 卡普

[FALSE] The run-time is O(nW) so it's pseudopolynomial
0-1 knapsack problem can be solved using dynamic programming in polynomial time

[FALSE] If neither of the two nodes are reachable by the negative cycle, the distance between them is calculated accurately.
The Bellman-Ford algorithm always fails to find the shortest path between two nodes in a graph if there is a negative cycle present in the graph.

[TRUE] Value of max flow = capacity of min-cut (iterate over all edges of the min-cut)
Given the min-cut, we can find the value of max flow in O(|E|).

[TRUE] Set the penalties to 0 and reward of 1 for every match
The sequence alignment algorithm can be used to find the longest common subsequence between two given sequences.

[FALSE] You only have to do it once.
In dynamic programming you must calculate the optimal value of a sub-problem twice, once during the bottom up pass and once during the top down pass.

[TRUE] For any cut which is not a min-cut this is True.
Maximum value of an s-t flow could be less than the capacity of a given s-t cut in a flow network.

[FALSE] Consider a graph with 4 nodes s, a, b and t and E = { (s,a), (a, t), (s, b), (b, t)}. The increase in the flow would be f + 2.
Suppose the maximum (s, t)-flow of some graph has value f. Now we increase the capacity of every edge by 1. Then the maximum (s, t)-flow in this modified graph will have value at most f +1.

[FALSE] The Orlin algorithm runs in O(mn)
There are no known polynomial-time algorithms to solve maximum flow.

[FALSE] Edges having capacity 1 does not mean |f| = 1 so the Ford-Fulkerson is still going to be pseudo-polynomial.
If all edges in a graph have capacity 1, then Ford-Fulkerson runs in linear time.

[FALSE] Choice of augmenting path can still affect the number of iterations of the inner loop.
In the scaled version of the Ford Fulkerson algorithm, choice of augmenting paths cannot affect the number of iterations.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[/FALSE]
A flow network with unique edge capacities has a unique min cut.

[TRUE/]
If a problem can be solved by dynamic programming, then it can always be solved by exhaustive search (Brute Force).

[TRUE/]
A divide and conquer algorithm acting on an input size of n can have a lower bound less than $\Theta(n \log n)$.

[/FALSE]
If a flow in a network has a cycle, this flow is not a valid flow.

[/FALSE]
In the divide and conquer algorithm to compute the closest pair among a given set of points on the plane, if the sorted order of the points on both X and Y axis are given as an added input, then the running time of the algorithm improves to O(n).

[TRUE/]
In a flow network, an edge that goes straight from *s* to *t* is always saturated when maximum *s* - *t* flow is reached.

[/FALSE]
The Bellman-Ford algorithm always fails to find the shortest path between two nodes in a graph if there is a negative cycle present in the graph.

[TRUE/]
If f is a max s-t flow of a flow network G with source s and sink t, then the capacity of the min s-t cut in the residual graph G_f is 0.

[/FALSE]
In a dynamic programming solution, the space requirement is always at least as big as the number of unique sub problems.

[/FALSE]
Decreasing the capacity of an edge that belongs to a min cut in a flow network may not result in decreasing the maximum flow.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE]
The Ford-Fulkerson Algorithm finds a maximum flow of a unit-capacity flow network with *n* vertices and *m* edges in time *O(mn)*.

[/FALSE]
In a flow network, if maximum flow is unique then min cut must also be unique.

[/FALSE]
In a flow network, if min cut is unique then maximum flow must also be unique.

[/FALSE]
In dynamic programming you must calculate the optimal value of a sub-problem twice, once during the bottom up pass and once during the top down pass.

[TRUE/]
Bellman-Ford algorithm solves the shortest path problem in graphs with negative cost edges in polynomial time.

[TRUE/]
The problem of deciding whether a given flow f of a given flow network G is a maximum flow can be solved in linear time.

[/FALSE]
An optimal solution to a 0/1 knapsack problem will always contain the object *i* with the greatest value-to-cost ratio V_i/C_i

[TRUE/]
The Ford-Fulkerson algorithm is based on greedy.

[TRUE/]
A flow network with unique edge capacities may have several min cuts.

[/FALSE]
Complexity of a dynamic programming algorithm is equal to the number of unique sub-problems in the solution space.

Q1:

FALSE

Suppose $f(n) = f\left(\binom{n}{2}\right)$+56, then $f(n) = \Theta(n)$

TRUE

Maximum value of an s-t flow could be less than the capacity of a given s-t cut in a flow network.

FALSE

For edge any edge e that is part of the minimum cut in G, if we increase the capacity of that edge by any integer k>1, then that edge will no longer be part of the minimum cut.

FALSE

Any problem that can be solved using dynamic programming has a polynomial worst case time complexity with respect to its input size. Not the input size

FALSE

In a dynamic programming solution, the space requirement is always at least as big as the number of unique sub problems

FALSE

In the divide and conquer algorithm to compute the closest pair among a given set of points on the plane, if the sorted order of the points on both X and Y axis are given as an added input, then the running time of the algorithm improves to O(n).

TRUE

If f is a max s-t flow of a flow network G with source s and sink t, then the value of the min s-t cut in the residual graph G_f is 0.

TRUE

Bellman-Ford algorithm can solve the shortest path problem in graphs with negative cost edges in polynomial time.

TRUE

Given a directed graph G=(V,E) and the edge costs, if every edge has a cost of either 1 or -1 then we can determine if it has a negative cost cycle in O(|V|^3) time.. Floyd-warshall

TRUE

The space efficient version of the solution to the sequence alignment problem (discussed in class), was a divide and conquer based solution where the divide step was performed using dynamic programming.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[/FALSE]
If an iteration of the Ford-Fulkerson algorithm on a network places flow 1 through an edge (u, v), then in every later iteration, the flow through (u, v) is at least 1.

A later augmenting path may pass through (v, u), causing the flow on (u, v) to be decreased.

[TRUE/]
For the recursion T(n) = 4T(n/3) + n, the size of each subproblem at depth k of the recursion tree is $n/3^{k-1}$.

[/FALSE]
For any flow network *G* and any maximum flow on *G*, there is always an edge *e* such that increasing the capacity of *e* increases the maximum flow of the network.

[/FALSE]
The asymptotic bound for the recurrence T(n) = 3T(n/9) + n is given by $\Theta(n^{1/2} \log n)$.

[/FALSE]
Any Dynamic Programming algorithm with *n* subproblems will run in *O(n)* time.

[/FALSE]
A pseudo-polynomial time algorithm is always slower than a polynomial time algorithm.

[TRUE/]
The sequence alignment algorithm can be used to find the longest common subsequence between two given sequences.

[/FALSE]
If a dynamic programming solution is set up correctly, i.e. the recurrence equation is correct and each unique sub-problem is solved only once (memoization), then the resulting algorithm will always find the optimal solution in polynomial time.

We don't know the time complexity of subproblem

[TRUE/]
For a divide and conquer algorithm, it is possible that the divide step takes longer to do than the combine step.

[TRUE/]
Maximum value of an $s-t$ flow could be less than the capacity of a given $s-t$ cut in a flow network.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/]
$T(n) = 9T(\frac{n}{5}) + n\log n$ then T(n) = $\theta(n^{\log_5 9})$

[/FALSE]
In the sequence alignment problem, the optimal solution can be found in linear time and space by incorporating the divide-and-conquer technique with dynamic programming.

[/FALSE]
Master theorem can always be used to find the complexity of T(n), if it can be written as the recurrence relation T(n) = aT($\frac{n}{b}$) + f(n).

[/FALSE]
In the divide and conquer algorithm to compute the closest pair among a given set of points on the plane, if the sorted order of the points on both X and Y axis are given as an added input, then the running time of the algorithm improves to O(n).

[TRUE/]
Maximum value of an s-t flow could be less than the capacity of a given s-t cut in a flow network.

[TRUE/]
One can **efficiently** find the maximum number of edge disjoint paths from s to t in a directed graph by reducing the problem to max flow and solving it using the Ford-Fulkerson algorithm.

[TRUE/]
In a network-flow graph, if the capacity associated with every edge is halved, then the max-flow from the source to sink also reduces by half

[TRUE/]
The time complexity to solve the max flow problem can be better than O(Cm) where C is the total capacity of the edges leaving the source and m is the number of edges in the network.

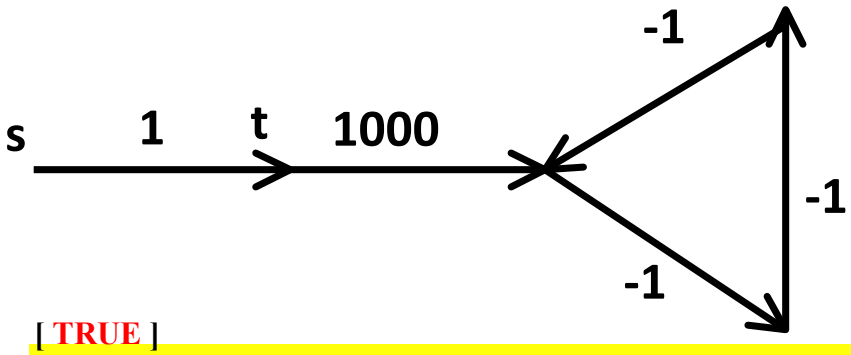
[TRUE/]
Bellman-Ford algorithm can handle negative cost edges even if it runs in a distributed environment using message passing.

[/FALSE]
If all edges in a graph have capacity 1, then Ford-Fulkerson runs in linear time.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[FALSE]
The Bellman-Ford algorithm always fails to find the shortest path between two nodes in a graph if there is a negative cycle present in the graph.

Counter example: Consider this graph. Bellman-Ford will limit the number of edges to at most 4 on the shortest path and terminates (because the number of nodes is 5). Thus, the shortest path from s to t is correct.



[TRUE]
A negative cycle in a graph can be detected in polynomial time.

The complexity is O(m*n) while m (number of edges) is upper-bounded by n^2, this this can be done in O(n^3), which is polynomial.

[FALSE]
In the sequence alignment problem, the optimal solution can be found in linear time and space by incorporating the divide-and-conquer technique with dynamic programming.

Linear in space only, not linear in time.

[TRUE]
The best time complexity to solve the max flow problem can be better than O(Cm) where C is the total capacity of the edges leaving the source and m is the number of edges in the network.

O(Cm) is the worst case. Therefore the best time complexity can be better than O(Cm).

[FALSE]
Bellman-Ford algorithm cannot solve the shortest path problem in graphs with negative cost edges in polynomial time.

If there is no negative cycle, the complexity is polynomial $O(n^3)$. If there is a negative cycle, the algorithm can also detect it in polynomial time $O(n^3)$. So the overall complexity is polynomial.

[TRUE]
In the Ford–Fulkerson algorithm, choice of augmenting paths can affect the number of iterations.

Choosing the paths with higher bottlenecks may significantly reduce the number of iterations.

[TRUE]
For flow networks such that every edge has a capacity of either 0 or 1, the Ford-Fulkerson algorithm terminates in O(n^3) time, where n is the number of vertices.

Complexity is O(Cm). However, if all capacities are either 0 or 1, then $C \leq n$ (number of nodes that are directly connected to source s). Thus, $Cm \leq nm \leq n^3$. Thus, Ford-Fulkerson will terminate in $O(n^3)$.

☆ **[TRUE]**
Every flow network with a non-zero max s-t flow value, has an edge e such that decreasing the capacity of e decreases the maximum s-t flow value.

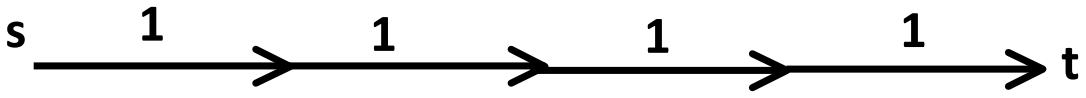
This is obvious from the max flow and min cut theorem.
Max flow $f^* = C(\text{min cut}) = C^*$. Decreasing the capacity of any edge that belongs to a min cut will result in a smaller cut $C' < C^*$, therefore the new max flow will be at most $C' < C^* = f^*$.

[TRUE]
Decreasing the capacity of an edge that belongs to a min cut in a flow network will always result in decreasing the maximum flow.

For the same reason as in the previous question.

[FALSE]
Every flow network with a non zero max s-t flow value, has an edge e such that increasing the capacity of e increases the maximum s-t flow value.

Counter example shown below: Increasing the capacity of any edge will not increase the max flow.



1) 20 pts
Mark the following statements as **TRUE**, **FALSE**. No need to provide any justification.

[FALSE] If f is a maximum s-t flow in a flow network G, then for all edges e out of s, we have $f(e) = c_e$.

[FALSE] Let (A, B) be a minimum s-t cut with respect to the capacities $\{ c_e : e \in E\}$. Now suppose we add 1 to every capacity, then (A,B) is still a minimum s-t cut with respect to these new capacities $\{1 + c_e : e \in E\}$.

[TRUE] If we multiply each edge with the same positive multiple "f", the max flow also gets multiplied by the same factor.

[TRUE] If a problem can be solved by dynamic programming, then it can always be solved by exhaustive search.

[TRUE] Sequence alignment problem between sequence X and sequence Y can be solved using dynamic programming in O(mn) when |X| = m and |Y| = n.

[FALSE] Bellman-Ford algorithm cannot solve the shortest path problem in graphs with negative cost edges in polynomial time.

[FALSE] If a dynamic programming solution is set up correctly, i.e. the recurrence equation is correct and the subproblems are always smaller than the original problem, then the resulting algorithm will always find the optimal solution in polynomial time.

[TRUE] In the Ford–Fulkerson algorithm, choice of augmenting paths can affect the number of iterations.

[FALSE] If in a flow network all edge capacities are distinct, then there exists a unique min-cut.

[TRUE] Max flow in a flow network can be found in polynomial time.

[TRUE]
Given a weighted, directed graph with no negative-weight cycles, the shortest path between every pair of vertices can be determined in worst-case time $O(V^3)$.

[FALSE]
Any problem that can be solved using dynamic programming has a polynomial time worst case time complexity with respect to its input size. **not respect to its input size**

[FALSE]
Ford-Fulkerson algorithm will always terminate as long as the flow network G has edges with strictly positive capacities. **irrational number will not result in termination**

[FALSE]
For any graph G with edge capacities and vertices ***s*** and ***t***, there always exists an edge such that increasing the capacity on that edge will increase the maximum flow from s to t. (Assume there is at least one path in the graph from ***s*** to ***t***)

[TRUE]
For any network and any maximum flow on this network, there always exists an edge such that decreasing the capacity on that edge with decrease the network's max flow.

[TRUE]
In the final residual graph constructed during the execution of the Ford–Fulkerson Algorithm, there's no path from source to sink.

[FALSE]
For edge any edge e that is part of the minimum cut in G, if we increase the capacity of that edge by any integer k>1, then that edge will no longer be part of the minimum cut.

[FALSE]
0/1 knapsack problem can be solved in polynomial time using dynamic programming **pseudo-polynomial**

[TRUE]
One can find the value of the optimal solution AND the optimal solution to the sequence alignment problem using only O(n) memory where n is the length of the smaller sequence.

[FALSE]
In a flow network if all edge capacities are irrational (not rational numbers) then the max flow is irrational.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

In the final residual graph constructed during the execution of the Ford–Fulkerson Algorithm, there's no path from source to sink.
TRUE

In a flow network whose edges all have capacity 1, the maximum flow value equals the maximum degree of a vertex in the flow network.
FALSE.

Memoization is the basis for a top-down alternative to the usual bottom-up approach to dynamic programming solutions.
TRUE

The time complexity of a dynamic programming solution is always lower than that of an exhaustive search for the same problem.
FALSE

If we multiply all edge capacities in a graph by 5, then the new maximum flow value is the original one multiplied by 5.
TRUE.

For any graph G with edge capacities and vertices s and t, there always exists an edge such that increasing the capacity on that edge will increase the maximum flow from s to t. (Assume that there is at least one path in the graph from $\boldsymbol{s}$ to $\boldsymbol{t}$.)
FALSE.
There might be more than one min-cut, by increasing the edge capacity of one min-cut doesn't have to impact the other one.

Let G be a weighted directed graph with exactly one source s and exactly one sink t. Let (A, B) be a maximum cut in G, that is, A and B are disjoint sets whose union is V, $s \in A, t \in B$, and the sum of the weights of all edges from A to B is the maximum for any two such sets. Now let H be the weighted directed graph obtained by adding 1 to the weight of each edge in G. Then (A, B) must still be a maximum cut in H.
FALSE
There could exist other edge which has many more cross-edges, which was not max-cut previously, to become the new max-cut.

A recursive implementation of a dynamic programming solution is often less efficient in practice than its equivalent iterative implementation.
We give points for both TRUE and FALSE answers due to the ambiguity of "often".

Ford-Fulkerson algorithm will always terminate as long as the flow network G has edges with strictly positive capacities.
FALSE
If some edges are irrational numbers, Ford-Fulkerson may never terminate.

Any problem that can be solved using dynamic programming has a polynomial time worst case time complexity with respect to its input size.
FALSE

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[FALSE]
If you have non integer edge capacities, then you cannot have an integer max flow.

[TRUE]
The maximum value of an s-t flow is equal to the minimum capacity of an s-t cut in the network.

[FALSE]
For any graph there always exists an edge such that increasing the capacity on that edge will increase the maximum flow from source s to sink t. (Assume that there is at least one path in the graph from source s to sink t.)

[FALSE]
If a problem can be solved using both the greedy method and dynamic programming, greedy will always give you a lower time complexity.

[FALSE]

There is a feasible circulation with demands $\{d_v\}$ if $\sum_v d_v = 0$. Reverse

[FALSE]
The best time complexity to solve the max flow problem is O(Cm) where C is the total capacity of the edges leaving the source and m is the number of edges in the network.

[TRUE]
In the Ford–Fulkerson algorithm, choice of augmenting paths can affect the number of iterations.

[FALSE]
0-1 knapsack problem can be solved using dynamic programming in polynomial time. pseudo-polynomial

[TRUE]
Bellman-Ford algorithm solves the shortest path problem in graphs with negative cost edges in polynomial time.

[FALSE]
If a dynamic programming solution is set up correctly, i.e. the recurrence equation is correct and the subproblems are always smaller than the original problem, then the resulting algorithm will always find the optimal solution in polynomial time.

may not polynomial time

[TRUE]
The problem of deciding whether a given flow f of a given flow network G is maximum flow can be solved in linear time. run BFS/DFS on residual graph and check whether there has any other path from s to t

[TRUE]
If you are given a maximum s - t flow in a graph then you can find a minimum s - t cut in time $O(m)$.

[TRUE]
An edge that goes straight from s to t is always saturated when maximum s - t flow is reached.

[FALSE]
In any maximum flow there are no cycles that carry positive flow.
(A cycle $<e_1, \ldots, e_k>$ carries positive flow iff $f(e_1) > 0, \ldots, f(e_k) > 0$.)

[TRUE]
There always exists a maximum flow without cycles carrying positive flow.

[FALSE]
In a directed graph with at most one edge between each pair of vertices, if we replace each directed edge by an undirected edge, the maximum flow value remains unchanged.

[TRUE]
The Ford-Fulkerson algorithm finds a maximum flow of a unit-capacity flow network (all edges have unit capacity) with n vertices and m edges in $O(mn)$ time.

[FALSE]
Any Dynamic Programming algorithm with n unique subproblems will run in $O(n)$ time. don't know anything about the time complexity of subproblems

[FALSE]
The running time of a pseudo polynomial time algorithm depends polynomially on the size of of the input

[FALSE]
In dynamic programming you must calculate the optimal value of a subproblem twice, once during the bottom up pass and once during the top down pass.

calculate once and memorize it.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

FALSE
If in a flow network all edge capacities are distinct, then there exists a unique min-cut. Could be more than one

FALSE
The Knapsack Problem describe in the textbook (0-1 Knapsack Problem) can be solved in a running time that is a polynomial time with respect to the input size, i.e. the number of objects (n). pseudo-polynomial time

TRUE
If a problem can be solved by dynamic programming, then it can always be solved by exhaustive search.

TRUE
Given a graph G(V, E) and its residual graph G_f(V', E'), then |V|=|V'| and 2|E|>=|E'|

FALSE
The Ford-Fulkerson Algorithm terminates when the source *s* is not reachable from the sink *t* in the residual graph G_f.

FALSE
If you have non integer edge capacities, then you cannot have an integer max flow.

FALSE
If *f* is a maximum *s*-*t* flow in *G*, where *s* is the source and *t* is the sink, then for all edges *e* out of *s*, we have $f(e) = c_e$.

TRUE
If the edge costs are all even, then the max flow (min cut) is also even.

FALSE
If the edge costs are all odd, then the max flow (min cut) is also odd. could be even

FALSE
If a problem can be solved using both the greedy method and dynamic programming, greedy will always give you a lower time complexity.

1) 20 pts
Mark the following statements as **TRUE**, **FALSE**. No need to provide any justification.

[TRUE]
If all capacities in a network flow are rational numbers, then the maximum flow will be a rational number, if exist.

[TRUE]
The Ford-Fulkerson algorithm is based on the greedy approach.

[FALSE]
The main difference between divide and conquer and dynamic programming is that divide and conquer solves problems in a top-down manner whereas dynamic-programming does this bottom-up.

[FALSE]
The Ford-Fulkerson algorithm has a polynomial time complexity with respect to the input size. **Ford-Fulkerson is pseudo polynomial**

[TRUE]
Given the Recurrence, $T(n) = T(n/2) + \theta(1)$, the running time would be $O(\log(n))$

[FALSE]
If all edge capacities of a flow network are increased by k, then the maximum flow will be increased by at least k. **could have more than one min-cut, and can be combined by difference number of edges**

[TRUE]
A divide and conquer algorithm acting on an input size of n can have a lower bound less than $\Omega(n \log n)$.

[TRUE]
One can actually prove the correctness of the Master Theorem.

[TRUE]
In the Ford Fulkerson algorithm, choice of augmenting paths can affect the number of iterations.

[FALSE]
In the Ford Fulkerson algorithm, choice of augmenting paths can affect the min cut.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

FALSE
For every node in a network, the total flow into a node equals the total flow out of a node. except for source and sink node

TRUE
Ford Fulkerson works on both directed and undirected graphs.

Because at every augmentation step you just need to find a path form s to t. This can be done in both direct and undriected graphs.

No
Suppose you have designed an algorithm which solves a problem of size n by reducing it to a max flow problem that will be solved with *Ford Fulkerson*, however the edges can have capacities which are $O(2^n)$. Is this algorithm efficient?

Yes
Is it possible for a valid flow to have a flow cycle (that is, a directed cycle in the graph, such that every edge has positive flow)?

positive flow cycles don't cause any problems. The flow can still be valid.

FALSE
Dynamic programming and divide and conquer are similar in that in each approach the sub-problems at each step are completely independent of one another.

FALSE
Ford Fulkerson has pseudo-polynomial complexity, so any problem that can be reduced to Max Flow and solved using *Ford Fulkerson* will have pseudo-polynomial complexity.

For example the edge disjoint paths problem is solved using FF in polynomial time. This is because for some problems the capacity C becomes a function of n or m.

TRUE
In a flow network, the value of flow from S to T can be higher than the number of edge disjoint paths from S to T.

FALSE
Complexity of a dynamic programming algorithm is equal to the number of unique sub-problems in the solution space.

TRUE
In *Ford-Fulkerson*'s algorithm, when finding an augmentation path one can use either BFS or DFS.

1) 10 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

TRUE
Binary search could be called a divide and conquer technique

FALSE
If you have non integer edge capacities, then you cannot have an integer max flow

TRUE
The Ford Fulkerson algorithm with real valued capacities can run forever

TRUE
If we have a 0-1 valued s-t flow in a graph of value f, then we have f edge disjoint s-t paths in the graph

TRUE
Merge sort works because at each level of the algorithm, the merge step assumes that the two lists are sorted

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

FALSE
For every node in a network, the total flow into a node equals the total flow out of a node. except for source and sink node

TRUE
Ford Fulkerson works on both directed and undirected graphs.

Because at every augmentation step you just need to find a path form s to t. This can be done in both direct and undriected graphs.

No
Suppose you have designed an algorithm which solves a problem of size n by reducing it to a max flow problem that will be solved with *Ford Fulkerson*, however the edges can have capacities which are $O(2^n)$. Is this algorithm efficient?

Yes
Is it possible for a valid flow to have a flow cycle (that is, a directed cycle in the graph, such that every edge has positive flow)?

positive flow cycles don't cause any problems. The flow can still be valid.

FALSE
Dynamic programming and divide and conquer are similar in that in each approach the sub-problems at each step are completely independent of one another.

FALSE
Ford Fulkerson has pseudo-polynomial complexity, so any problem that can be reduced to Max Flow and solved using *Ford Fulkerson* will have pseudo-polynomial complexity.

For example the edge disjoint paths problem is solved using FF in polynomial time. This is because for some problems the capacity C becomes a function of n or m.

TRUE
In a flow network, the value of flow from S to T can be higher than the number of edge disjoint paths from S to T.

FALSE
Complexity of a dynamic programming algorithm is equal to the number of unique sub-problems in the solution space.

TRUE
In *Ford-Fulkerson*'s algorithm, when finding an augmentation path one can use either BFS or DFS.