**[ TRUE/FALSE ] 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.

**[ TRUE/FALSE ] 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/FALSE ] TRUE**

An edge that goes straight from *s* to *t* is always saturated when maximum *s - t* flow is

reached.

**[ TRUE/FALSE ] FALSE**

In any maximum flow there are no cycles that carry positive flow.

(A cycle <e1, …, ek> carries positive flow iff f(e1) > 0, …, f(ek) > 0.)

**[ TRUE/FALSE ] TRUE**

There always exists a maximum flow without cycles carrying positive flow.

**[ TRUE/FALSE ] 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/FALSE ] FALSE**

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.

**[ TRUE/FALSE ] FALSE**

Any Dynamic Programming algorithm with *n* unique subproblems will run in *O(n)*

time.

**[ TRUE/FALSE ] FALSE**

The running time of a pseudo polynomial time algorithm depends polynomially on

the size of of the input

**[ TRUE/FALSE ] 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.

1. 20 pts  
   For each of the following statements, answer whether it is TRUE or FALSE, and briefly justify your answer.
2. If a connected undirected graph G has the same weights for every edge, then every spanning tree of G is a minimum spanning tree, but such a spanning tree cannot be found in linear time.

True. Just do a BFS

1. Given a flow network G and a maximum flow of G that has already been computed, one can compute a minimum cut of G in linear time.

False. You need the residual graph. Just the value of flow wont do

1. The Ford-Fulkerson Algorithm finds a maximum flow of a unit-capacity flow network with n vertices and m edges in time O(mn) if one uses depth-first search to find an augmenting path in each iteration.

True. Given in the book

1. Unless P = NP, 3-SAT has no polynomial-time algorithm.

This is true.

1. 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.

True If you are given a residual graph, then you can do a BFS and see if there is a path from source to sink

1. If a decision problem A is polynomial-time reducible to a decision problem B (i.e., A≤ pB ), and B is NP-complete, then A must be NP-complete.

This is False. A could be a polynomial time solvable problem, which is always reducible to the NP complete problem in polynomial time.

1. If a decision problem B is polynomial-time reducible to a decision problem A (i.e., B≤ pA ), and B is NP-complete, then A must be NP-complete.

This is False. Recall that the reduction only proves NP hardness. In this case, we can only say that A is NP hard. We have yet to prove that it is in NP, which requires giving a certificate and certifier. NP hardness + membership in NP = NP completeness

1. Integer max flow ( where flows and capacities are integers) is polynomial time reducible to linear programming .

If integer max flow was reducible in polynomial time to linear programming, then linear programming would be NP hard since Integer Programming is NP hard

1. It has been proved that NP-complete problems cannot be solved in polynomial time.

False

1. NP is a class of problems for which we do not have polynomial time solutions.

No. It is the class of problems for which we have a poly size certificate and poly time certifier

**[ 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.

**[ TRUE ]** Given the Recurrence, *T*(*n*) = *T*(*n/*2 ) + θ(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.

**[ TRUE ]**

A divide and conquer algorithm acting on an input size of n can have a lower bound

less than Ω(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.

True **[ TRUE/FALSE ]**

Binary search could be called a divide and conquer technique

False **[ TRUE/FALSE ]**

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

**Ture [ TRUE/FALSE ]**

The Ford Fulkerson algorithm with real valued capacities can run forever

**Ture [ TRUE/FALSE ]**

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

paths in the graph

**Ture [ TRUE/FALSE ]**

Merge sort works because at each level of the algorithm, the merge step assumes that

the two lists are sorted

**[ TRUE/FALSE ]** True

Max flow problems can in general be solved using greedy techniques.

**[ TRUE/FALSE ]** False

If all edges have unique capacities, the network has a unique minimum cut.

**[ TRUE/FALSE ]** True

Flow f is maximum flow if and only if there are no augmenting paths.

**[ TRUE/FALSE ]** True

Suppose a maximum flow allocation is known. Increase the capacity of an edge by 1 unit. Then, updating a max flow can be easily done by finding an augmenting path in the residual flow graph.

**[ TRUE/FALSE ]** False

In order to apply divide & conquer algorithm, we must split the original problem into at least half the size.

**[ TRUE/FALSE ]** True

If all edge capacities in a graph are integer multiples of 5 then the maximum flow value is a multiple of 5.

**[ TRUE/FALSE ]** False

If all directed edges in a network have distinct capacities, then there is a unique maximum flow.

**[ TRUE/FALSE ]** True

Given a bipartite graph and a matching pairs, we can determine if the matching is maximum or not in O(V+E) time

**[ TRUE/FALSE ]** False

Maximum flow problem can be efficiently solved by dynamic programming

**[ TRUE/FALSE ] True**

**The difference between dynamic programming and divide and conquer techniques is that in divide and conquer sub-problems are independent**