2018 Fall

[**TRUE/FALSE**] Dynamic programming only works on problems with non-overlapping subproblems.

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

[**TRUE/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.

[**TRUE/FALSE**] A dynamic programming solution always explores the entire search space for all possible solutions.

[**TRUE/FALSE**] Decreasing the capacity of an edge that belongs to a min cut in a flow network always results in decreasing the maximum flow.

[**TRUE/FALSE**] Suppose f is a flow of value 100 from s to t in a flow network G. The capacity of the minimum cut s-t in G is equal to 100.

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

[**TRUE/FALSE**] Given a flow network where all the edge capacities are even integers, the algorithm will require at most C/2 iterations, where C is the total capacity leaving the sources.

[**TRUE/FALSE**] By combining divide and conquer with dynamic programming we were able to reduce the space requirements for our sequence alignment solution at the cost of increasing the computational complexity of our solution.

2017 Fall

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

[**TRUE/FALSE**] In a connected, directed graph with positive edge weights, the Bellman-Ford algorithm runs asymptotically faster than the Dijkstra algorithm.

[**TRUE/FALSE**] There exist some problems that can be solved by dynamic programming but cannot be solved by greedy algorithm.

[**TRUE/FALSE**] The Floyd-Warshall algorithm is asymptotically faster than running the Bellman-Ford algorithm from each vertex.

[**TRUE/FALSE**] If we have a dynamic programming algorithm with n^2 subproblems, it is possible that the space usage could be $\mathrm{O(n)}$.

[**TRUE/FALSE**] The Ford-Fulkerson algorithm solves the maximum bipartite matching problem in polynomial time.

[**TRUE/FALSE**] Given a solution to a max-flow problem, that includes the final residual graph G_f. We can verify in a linear time that the solution does indeed give a maximum flow.

[**TRUE/FALSE**] In a flow network, a flow value is upper-bounded by a cut capacity.

[**TRUE/FALSE**] In a flow network, a min-cut is always unique.

[**TRUE/FALSE**] A maximum flow in an integer capacity graph must have an integer flow on each edge.

2017 Summer

[TRUE/FALSE]
Given the value of max flow, we can find a min-cut in linear time.

[TRUE/FALSE]
Given a min-cut, we can find the value of max flow in linear time.

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

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

[TRUE/FALSE]
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.

[TRUE/FALSE]
The Floyd-Warhsall algorithm always detects if a graph has a negative cycle.

[TRUE/FALSE] Increasing the capacity of an edge that belongs to a min-cut in a flow network may not result in increasing the maximum flow.

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

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

[TRUE/FALSE] We can use the Bellman-Ford algorithm for undirected graph with negative edge weights.

2017 Spring

[TRUE/FALSE] Given the value of max flow, we can find a min-cut in linear time.

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

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

[TRUE/FALSE] 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.

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

[TRUE/FALSE] 0/1 knapsack problem can be solved using dynamic programming in polynomial time, but not strongly polynomial time.

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

[TRUE/FALSE] The Travelling Salesman problem can be solved using dynamic programming in polynomial time.

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

[TRUE/FALSE] We can use the Bellman-Ford algorithm for undirected graph with negative edge weights.

2016 Fall

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

[**TRUE/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/FALSE**] Given the min-cut, we can find the value of max flow in O(|E|).

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

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

[**TRUE/FALSE**] 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.

[**TRUE/FALSE**] There are no known polynomial-time algorithms to solve maximum flow.

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

[**TRUE/FALSE**] In the scaled version of the Ford Fulkerson algorithm, choice of augmenting paths cannot affect the number of iterations.

2016 Spring

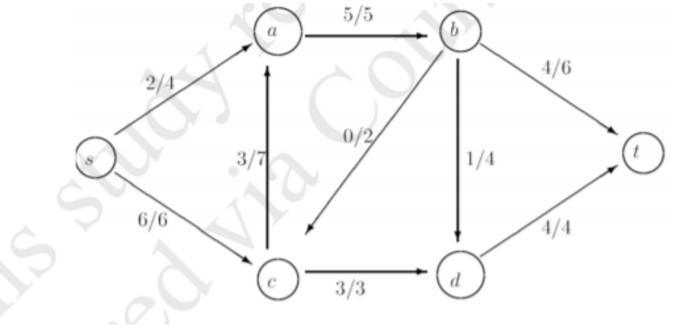
[TRUE/FALSE] The number of iterations it takes Bellman-Ford to converge can vary depending on the order of nodes updated within an iteration.

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

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

[TRUE/FALSE] The following flow is a maximal flow.



Note: The notation a/b describes a units of flow on an edge of capacity b.

[TRUE/FALSE] In a circulation network, there is a feasible circulation with demands {dv}, if $\sum v\ dv$ = 0

[TRUE/FALSE] An optimal solution to a 0/1 knapsack problem will always contain the object i with the greatest value-to-cost ratio Vi/Ci.

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

[TRUE/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/FALSE] 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.

2015 Fall

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

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

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

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

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

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

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

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

2015 Spring

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

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

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

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

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

[TRUE/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/FALSE]
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 Gf is 0.

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

[TRUE/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.

2014 Fall

[TRUE/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.

[TRUE/FALSE]

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

[TRUE/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.

[TRUE/FALSE]

The asymptotic bound for the recurrence T(n) = 3T(n/9) + n is given by Θ(n1/2 log n).

[TRUE/FALSE]

Any Dynamic Programming algorithm with n subproblems will run in O(n) time.

[TRUE/FALSE]

A pseudo-polynomial time algorithm is always slower than a polynomial time algorithm.

[TRUE/FALSE]

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

[TRUE/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.

[TRUE/FALSE]

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

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

2013 Summer

[**TRUE/FALSE**] If a network with a source s and sink t has a unique max s-t flow, then it has a unique min s-t cut.

[**TRUE/FALSE**] 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.

[**TRUE/FALSE**] Fractional knapsack problem can be solved in polynomial time.

[**TRUE/FALSE**] Subset-sum problem can be solved in polynomial time

[**TRUE/FALSE**] 0-1 knapsack problem can be solved in polynomial time

[**TRUE/FALSE**] Max flow in a flow networks can be found in polynomial time

[**TRUE/FALSE**] Ford-Fulkerson algorithm runs in polynomial time

[**TRUE/FALSE**] One can determine if a flow is valid or not in linear time with respect to the number of edges and nodes in the graph.

[**TRUE/FALSE**] Given the value of flow v(f) for a flow network, one can determine if this value is the maximum value of flow or not in linear time.

[**TRUE/FALSE**] A dynamic programming algorithm always uses some type of recurrence relation.

2011 Summer

[TRUE/FALSE] Dynamic programming is a brute force method.

[TRUE/FALSE] Max flow in a flow network with real valued capacities can be found in polynomial time.

[TRUE/FALSE] The top down pass in dynamic programming produces the **value** of the optimal solution whereas the bottom up pass produces the actual solution.

[TRUE/FALSE] If a dynamic programming solution is formulated correctly, then it can be solved in polynomial time.

[TRUE/FALSE] A flow network can have many sources and sinks but only one super source and one super sink.

[TRUE/FALSE] It is possible to have a max flow of zero in a flow network even if all edge capacities are greater than zero.

[TRUE/FALSE] If the capacities of edges in a flow network G are changed and the total increase of all edge capacities is k, then the value of a maximum flow of G increases by at most k.

[TRUE/FALSE] For a flow network G = (V, E) that can carry a flow of up to $k>0$ units, if the capacity of each edge in a flow network is multiplied by m, then the resulting flow network will have a max flow of value mk.

[TRUE/FALSE] with justification
Although the original Ford-Fulkerson algorithm may fail to terminate in the case where edge capacities are arbitrary real numbers, it can be slightly modified so that it is guaranteed to terminate in the case where edge capacities are rational numbers, regardless of how the augmenting paths are chosen. This is an immediate consequence of the statement in Question 3.

2010 Summer

[TRUE/FALSE] 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).

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

[TRUE/FALSE] Ford-Fulkerson algorithm will always terminate as long as the flow network G has edges with strictly positive capacities.

[TRUE/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/FALSE] 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/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] 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.

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

[TRUE/FALSE] 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.

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

2009 Spring

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

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

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

[**TRUE/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**] 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 $\mathrm{O(mn)}$ time.

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

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

[**TRUE/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.

2008 Spring

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

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

[**TRUE/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.

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

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

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

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

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

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