1. _**True**_To use the greedy method, a problem should have the property that a series of locally optimal choices lead to a global optimal configuration.
2. _**False**_The greedy approach should never be used on optimization problems.
3. _**False**_The task scheduling problem is an example of the divide-and-conquer method because tasks are sorted by their running time before assigning them to machines.
4. _**True**_The fractional knapsack problem can be solved using the greedy method by continued selection of the item with the highest benefit-to-weight ratio.
5. _**True**_Recurrence equations are used to evaluate the time-complexity of divide-and-conquer algorithms.
6. ______The problem of multiplying big integers of size n has O ($n^{1.585}$) time-complexity.
7. _**True**_The dynamic programming technique is similar to the divide-and-conquer approach in the way that a problem is divided in smaller, independent sub-problems and the results merged together to form the solution.
8. ______When the dynamic programming technique is applied to the multiplying matrices problem the time-complexity is reduced from exponential to linear.

9. _**False**_Dynamic programming algorithms have a running time that only depends on n.
10. _**True**_Two vertices that are adjacent are endpoints of the same edge.
11. _**False**_The sum of the degrees of all vertices in a graph G are equal to the number of edges.
12. _**False**_ A spanning tree of a graph contains only some of the vertices of the graph.
13. _**True**_An adjacency list structure has similar performance to the edge list structure but also has performance improvements in methods such as incidentEdges(v) and areAdjacent(u,v).

14. _**True**_Depth-first search traversal of an undirected graph uses the backtracking technique to explore the vertices and edges of a graph.
15. _**False**_In breadth-first search, the edges are marked as either discovery or cross edges to indicate their role in the spanning tree.
16. _**False**_To test whether a graph is connected, a DFS traversal can be performed and if some of the vertices are not marked as discovered, the graph is not connected.
17. ______In a computer network, reachability in a directed graph is computed to find out if a message can be routed from node v to node w.

18. ______The transitive closure of a graph measures the density of the edges in the graph.
19. ______A topological ordering of a digraph is useful in scheduling tasks that have constraints as represented in the graph.
20. _**False**_Single-source shortest path algorithms find all the paths between a vertex v and w in a weighted graph.

Multiple choice questions. Pick the best answer. [3 points each]
21. The depth-first search algorithm we studied uses ____.
a) edge reversal   b) a min-heap   c) **recursion**   d) transitive closure

22. The ____ problem can be solved optimally with the greedy approach.
a) sum-of-subsets
b) traveling salesman
c) big integer multiplication
d) fractional knapsack

23. The iterative substitution method is a technique that depends on our ability to ____ that can be converted to the closed-form version of the recurrence equation.
a) see a pattern
b) multiply matrices
c) draw a tree
d) apply a formula

24. In the following recurrence relation, the step of merging sub-problem solutions is done ___ times at each level.

$$T(n) = \begin{cases} 5b & \text{if } n < 2 \\ 3T(n/2) + 4n & \text{if } n \geq 2 \end{cases}$$

a) 2 b) 3 c) 4 d) 5

25. The ________ applies to a problem if the best solution always contains optimal solutions of all sub-problems.
a) principle of optimality
b) greedy method
c) iterative method
d) big O notation

26. Which data structure is preferred when we need space efficiency when representing a graph with 10,000 vertices and 200,000 edges and we also need fast response to the areAdjacent method?
a) adjacency matrix
b) adjacency list
c) edge list

27. Which of the following problems has not been proven to be intractable, but also does not have a polynomial-time solution?
a) 0-1 knapsack problem
b) minimum spanning tree problem
c) matrix multiplication problem
d) searching problem

28. What is an application of topological ordering?
a) sorting mail by the user's name
b) finding the path between a pair of vertices
c) determining connectivity in a graph
d) showing the inheritance hierarchy in Java interfaces

29. A weighted graph will only have one minimal spanning tree if ______.
a) every edge has a different weight
b) every edge has the same weight
c) every vertex connects to very other vertex
d) every vertex is a separate component

30. A _____ is a sequence of vertices that have an edge between each vertex and its successor.
a) cycle
b) map
c) component
d) path

31. Problems that are in NP can be ______ in polynomial time.
a) solved
b) **verified**
c) analyzed
d) halted

32. Some dynamic programming algorithms that we studied have a worst-case time complexity that is _____ , where the running time depends on the magnitude of a number in the input.
a) exponential
b) quadratic
c) differential
d) pseudo-polynomial

Asymptotic Performance

◆ n vertices, m edges ◆ no parallel edges ◆ no self-loops ◆ Bounds are "big-Oh"	Edge List	Adjacency List	Adjacency Matrix
aVertex()	1	1	1
edges()	m	m	m
vertices()	n	n	n
endVertices(e)	1	1	1
opposite(v, e)	1	1	1
degree(v)	m	1	n
numEdges()	1	1	1

Asymptotic Performance

◆ n vertices, m edges ◆ no parallel edges ◆ no self-loops ◆ Bounds are "big-Oh"	Edge List	Adjacency List	Adjacency Matrix
Space	$n + m$	$n + m$	n^2
incidentEdges(v)	m	$\deg(v)$	n
areAdjacent(v, w)	m	$\min(\deg(v), \deg(w))$	1
insertVertex(o)	1	1	n^2
insertEdge(v, w, o)	1	1	1
removeVertex(v)	m	$\deg(v)$	n^2
removeEdge(e)	1	1	1

1. What is the relationship between the incident edges of a vertex v and the degree of v?

   Ans: **Number of incident edges of a vertex v equals with degree of vertex v**

   incidentEdges(v).count() = deg(v)

2. A path is an alternating sequence of **vertices** and **edges** that starts with a **vertex** and ends with a different **vertex**.

3. A cycle is an alternating sequence of **vertices** and **edges** starting with a **vertex** and ending with the same **vertex.**

4. If the graph G is implemented using adjacency lists, then the running time for G.incidentEdges(v) is **deg(v)**. Do not include constants or low order terms, e.g., 2m or

O(m) should be written simply as m and n-1 should be written as n otherwise the automatic grader will mark the answer as wrong.

5. If the graph G is implemented using an adjacency matrix, then the running time for G.incidentEdges(v) is **n**. Do not include constants or low order terms, e.g., 2m should be written m and n-1 or O(n) should be written n otherwise the automatic grader will mark the answer as wrong.

6. If the graph G is implemented using adjacency lists, then the running time for G.areAdjacent(u,v) is the minimum of the **deg(v)** and **deg(w)**. Do not include constants or low order terms, e.g., 2m or O(m) or m-1 should be simply written m, otherwise the automatic grader will mark the answer as wrong

7. If the graph G is implemented using an adjacency matrix, then the running time for G.areAdjacent(v) is **1**. Do not include constants or low order terms, e.g., 2m should be written m and n-1 or O(n) should be written n otherwise the automatic grader will mark the answer as wrong.

8. What is the difference between a simple path and a non-simple path?

Simple path is all vertices and edges are distinct. Not duplicated. Non-simple path can contains duplicated vertices

9. A subgraph is subset of the edges and vertices such that none of the edges connect to any of the **vertices** outside of this subgraph.

10. What is the difference between a subgraph and a spanning subgraph?

Spanning subgraph must contain all vertices of main graph G. Subgraph cannot contain some vertices

11. Two **vertices** are connected if there is a **path** between them. A graph is connected if all **vertices** are connected.

12. A connected component is the set of all vertices that are **connected** and all the edges that **connected** these vertices.

13. A tree T is a graph that is **connected** and **no cycles**.

14. The connected components of a forest are **trees**. None of the components of a forest have any **cycles**. All of the vertices in a component of a forest are **connected**.

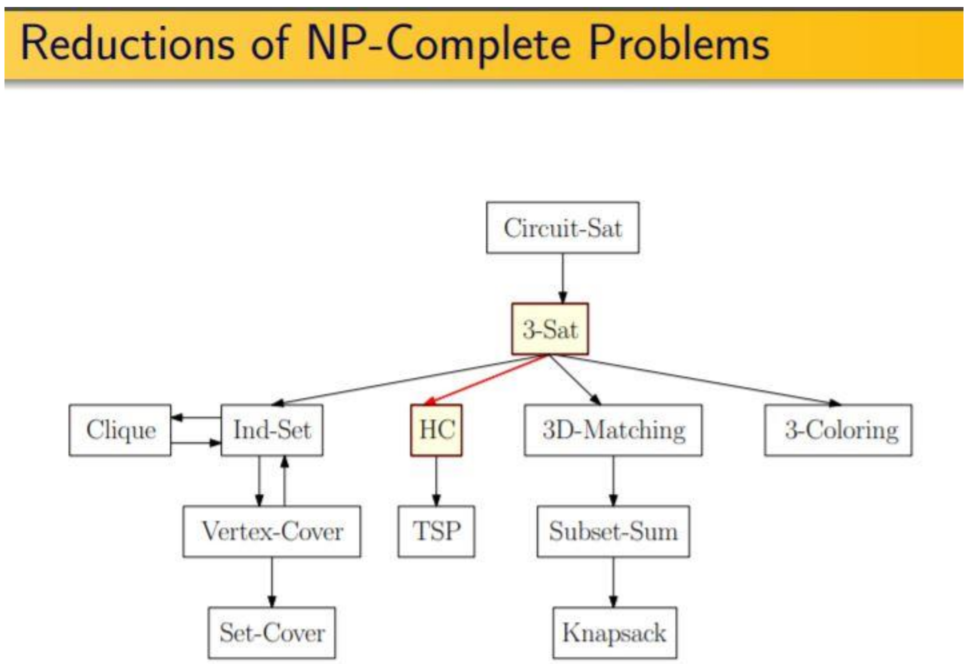
15. Suppose A -> B (A can be reduced to B in polynomial time). Suppose also that B is in NPC. If A can be shown to be a member of P, then we have shown that P=NP. If true, explain why. If false, give a counter example or explain why.

False , if P=NP then B should be reduced A.

16. Suppose A -> B (A can be reduced to B in polynomial time). If both A and B are in NPC, then B -> A (B can be reduced to A in polynomial time). If true, explain why. If false, give a counter example or explain why.
True, A should be equal or at least harder than B. So B and A is equal.

17. Which of the following problems is intractable, but could become tractable if someone finds/designs a new algorithm? Select all for which this applies.

Easy (P, tractable)
hard (NPH, NPC, intractable)
noncomputable (NPH, undecidable)



18. If problem A can be reduced to problem B in polynomial time, then what is the relative difficulty of A and B?

19. We say that problem A is NP-hard, if **every problem in NP** can be reduced to A in polynomial time.

20. How do we prove that a problem A is a member of NP-Hard?

Every problem A ϵ NP can be reduced to Q in polynomial-time

21. As long as P is not equal to NP, what can we say/conclude will continue to be true about a problem A if A is a member of NPC? I'm looking for the practical useful conclusions for you in your career, especially if you need to do something for your boss related to computing a solution to an NPC problem and you have a deadline of some kind.
    First we need to prove that NPC. In addition, work on an approximation algorithm.