1. In the following example, what is the value of the max-weight independent set, and that of the output of our greedy algorithm

1-4-5-4

1. 14 and 10
2. 8 and 6
3. 8 and 8
4. 9 and 8

(2) is the correct answer. The maximum indepedent set are the vertices with weights 4 and 4. The greedy algorithm would select the largest weight vertex and from there select any maximum weight vertex that is still valid. This would result in a total weight of 6. This shows that a greedy approach the problem will not be optimal

2. How many different independent sets does a complete graph with 5 vertices have? How about a cycle with 5 vertices?

1. 1 and 2
2. 5 and 10
3. 6 and 11
4. 6 and 16

(3) is the correct answer. For the complete graph once we have chosen a vertex we have no other vertex that can be joined alongside it. Since we have 5 choices the amount of independent sets with at least one vertex is 5. We now include the independent set with no vertices for a total of 6 independent sets for a graph complete graph with 5 vertices. The independent sets are as folowed $\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \emptyset$ For a cycle graph with 5 vertices we can again use the same analysis to get 6 independent sets but now we have the options of getting independent sets with at least two members. The total number of independent sets with at least two elements is 5 hence we get a total of 11 independent sets for a cycle graph with 5 vertices. The empty sets are as followed $\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \emptyset, \{v_1, v_4\}, \{v_1, v_3\}, \{v_5, v_2\}, \{v_5, v_3\}, \{v_4, v_2\}$.

3. What is the total weight of the output of the greedy algorithm when the input graph is the four-vertex path? Is this the maximum possible?

1. 6;no
2. 6;yes

3. 8;no

4. 8;yes

(1) is the correct answer. The greedy algorithm would first choose the 5 and then choose the 1 to get a total weight of 6. But the maximum independent set has a total weight of 8 implying that the greedy algorithm fails to solve the problem correctly.

4. What is the asymptotic running time of the recursive *WIS* algorithm, as a function of the number n of vertices?

1. $O(n)$
2. $O(n \cdot logn)$
3. $O(n^2)$
4. none of the above

(4) We have a recurrence of the form $T(n) = T(n-1) + T(n-2) + \Theta(1) \geq F_n$ where F_n is nth fibonacci number. This simplifies to $T(n) \geq \Theta(\phi^n)$ where ϕ is the golden ratio.

5. Each of the recursive calls of the recursive WIS algorithm is responsible for computing an MWIS of a specified input graph. Ranging over all of the calls, how many *distinct* input graphs are ever considered?

1. $\Theta(1)$
2. $\Theta(n)$
3. $\Theta(n^2)$
4. $2^{\Theta(n)}$

(2) is the correct answer we compute the values for $G_n, G_{n-1} \ldots, G_0$ which in total is $n+1$ graphs. This is order of n and hence is $\Theta(n)$.

6. Consider an instance of the knapsack problem with knapsack capacity $C = 6$ and four items $(3,4), (2,3), (4,2), (4,3)$ where the first element in the ordered pair denotes the value and the second element denotes the size.

1. 6
2. 7

3. 8

4. 10

(3) is the correct answer. If we choose the the values of 4 we'll fill our knapsack with total value of 8.

7. Which of the following statemens hold for the set $S - \{n\}$?

1. It is an optimal solution to the subproblem consisting fo the first $n-1$ items and knapsack capacity C.

2. It is an optimal solution to the subproblem consisting of the first $n-1$ items and knapsack capacity $C - v_n$

3. It is an optimal solution to the subproblem consisting of the first $n-1$ items and knapsack capacity $C - s_n$.

4. It might not be feasible if the knapsack capacity is only $C - s_n$.

(c) is the correct answer.

8. Consider the input graph

5-3-1-7-2-4-6

where vertices are labeled with their weights. What are the final array entries of the *WIS* algorithm and which vertices belong to the *MWIS*?

The entries of the array are [0, 5, 5, 6, 12, 12, 16, 18]. the vertices that make up the MWIS are v_0, v_3 and v_6.

9. Which of the following statements hold?

1. The *WIS and WIS Reconstruction* algorithm always return a solution that includes a maximum weight vertex

2. When vertices' weights are distinct, the *WIS and WIS Reconstruction* algorithms never return a solution that included a minimum-weight vertex

3. If a vertex v does not belong to an *MWIS* of the prefix G_i comprising the first i vertices and $i-1$ edges of the input graph, it does not belong to any *MWIS* of $G_{i+1}, G_{i+2}, \ldots, G_n$ either.

4. If a vertex v does not belong to an *MWIS* of G_{i-1} or G_i, it does not belong to any MWIS of $G_{i+1}, G_{i+2}, \ldots, G_n$ either.

(1) is false. problem (1) shows a solutions that does not include a maximum-weight vertex. (2) is false the path graph 1-3-9 will include the vertex with weight 1 in its set. (3) is incorrect. (4) is correct. We can use lemma 16.1 to argue that if it was a vertex v was not in the $MWIS$ of G_{i-1} or G_{i-2} then it will not be included in further prefixes of G_i.

10. This problem outlines an approach to solving the WIS problem in graphs more complicated than paths. Consider an arbitrary graph $G = (V, E)$ with nonnegative vertex weights, and an arbitrary vertex $v \in V$ with weights w_v. Obtain H from G by removing v and its incident eges from G. Obtain K from H by removing $v's$ neighbors and their incident edges. Let W_G, W_H, and W_K denote the total weight of an MWIS in G, H and K, respectively, and consider the formula

$$W_G \quad = \quad max\{W_H, W_K + W_v\}$$

Which of the following statements are true?

1. The formula is not always correct in path graphs
2. The formula is always correct in path graphs but now always in trees
3. The formula is always correct in trees but not always correct in arbitrary graphs
4. The formula is always correct in arbitrary graphs
5. The formula leads to a linear-time algorithm for the WIS problem in trees.
6. The formula leads to a linear-time algorithm for the WIS problem in arbitrary graphs.

11. Consider an instance of the knapsack problem with five items $(1, 1), (2, 3), (3, 2), (4, 5), (5, 4)$ where the first element denotes the value and the second denotes the weight and knapsack capacity $C = 9$. What are the final array entries fo the Knapsack algorithm and which items belong to the optimal solution?

The final solutions to the knapsack problem for this instance is 10, the elements at index $1, 2$ and 4 are the elements chosen.