Question 1 (1 marks)
What property must be true between a parent node and its children in a binary search tree.

Answer: Parent must be greater or equal to the left child, and right child must be greater than or equal to the parent.

Question 2 (1 marks)
What is the time complexity for search in a binary search tree, and for search in a doubly linked list.

Answer: Binary search tree is \(O(h)\), here \(h\) is the height of the tree, which should be \(\log n\). Doubly linked list is \(O(n)\).

Question 3 (1 marks)
For a hash table, how long does search take where collisions are resolved by chaining? What assumption did you have to make? Explain what the assumption is rather than just give its name.

Answer: \(1 + \alpha\) where \(\alpha\) is the load factor. Assumed simple uniform hashing, where you assume that the probability of where a key hashes to does not depend on where the previous keys hashed to.

Question 4 (1 marks)
For a hash table with open addressing (storing the keys in the table), how are collisions resolved? Give a method for doing this.

Answer: You need to probe other positions in the table. Need to make sure you can probe every location in the table.

Question 5 (1 marks)
Answer one of the following questions:
1. Describe how deletion is handled with open addressing.
2. Describe the assumption for open addressing that guarantees search is done in \(1/(1 - \alpha)\)

Answer: Need to leave a marker. For search, assume marker means it is full, and for insertion means spot is empty.

Need to assume that the probe sequence is equally likely to be any of the \(m!\) permutations.

Question 6 (1 marks)
What operation does a min priority queue allow you to do in constant time? Give an operation that a min priority queue does not support? In other words, what operations takes \(O(n)\) time, that some other data structure can do more quickly.

Answer: Find the minimum
Find the maximum, search, next, prev, print

Question 7 (1 marks)
Disjoint sets is where you need to represent what set each element is in, where an element is in exactly one set. Give two approaches for representing a disjoint set, and describe each in a sentence or two (rather than with just a phrase). Which one allows all of the members of a disjoint set to be determined?
**Answer:** Linked lists: each set has a linked list that indicates its elements. You can enumerate through the elements in a set.

Rooted tree: elements for a set are stored in a tree, with just parent pointers. Since there are no child pointers, you cannot determine the children of a node, so you cannot enumerate through the elements.

---

**Question 8** (1 marks)
One of the implementations can use the heuristic of path compression and union by rank. Choose one of these heuristics, and explain how it is implemented and why it is supposed to speed up execution time.

**Answer:**

**Question 9** (1 marks)
Here is the definition of $O(n)$:

$$O(g(n)) = \{ f(n) : \text{there exists positive constants } c \text{ and } n_0 \text{ s.t. } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \}$$

Prove that $2n + 5 \in O(n)$ using the definition.

**Answer:**

**Question 10** (1 marks)
Give the definition for $\Theta$

**Answer:**

**Question 11** (1 marks)
Explain why $\Theta$ might not be defined for an algorithm, but $O$ and $\Omega$ are defined.

**Answer:** An algorithm like search in a non sorted list has time $O(n)$ versus $\Omega(1)$. Since these are not the same, $\Theta$ is not defined.

**Question 12** (1 marks)
Arrange the following by their relative rate of growth, from smallest to largest. $n^3$, $2^n$, $\sqrt{n}$, $n \log n$, $n^2$, $\log n$, 1, $n$.

**Answer:**

**Question 13** (1 marks)
Explain what optimal substructure means.

**Answer:**

**Question 14** (2 marks)
In the rod-cutting problem, you need to find the best cuts of a rod of length $n$ to maximum your revenue $r_n$, given table $p$, where $p_i$ is the price you will receive for a rod of length $i$.

Give an equation that gives the maximum revenue for cutting a rod of length $n$ ($r_n$) in terms of the maximum revenue for cutting a rod of smaller length ($r_i$).

**Answer:**
Question 15  (1 marks)
Here is the recursive equation for the binary knapsack problem.
Let \( b(i, j, w) \) be the best value of a subset of the items from \( i \) through \( j \) inclusive whose weight is at most \( w \).

\[
b(0, 0, w) = \begin{cases} 
  v_0 & \text{if } w_0 \leq w \\
  0 & \text{otherwise}
\end{cases}
\]

For \( i > 0 \)

\[
b(0, i, w) = \max(b(0, i-1, w), b(0, i-1, w-w_i) + v_i)
\]

Explain how to solve this with dynamic programming with a bottom-up approach. Make sure you explain why subproblems are not repeated solved, and how you need to order the subproblems.

Answer:

Question 16  (1 marks)
In chapter 6 on Heapsort, the textbook explains how to view a binary tree as an array where array[0] is the top of the tree. Write the array with values [23,17,14,6,13,10,1,5,7,12] as that binary tree. Is it in max-heap form?

Answer:

```
23
  
  17 14
  
  6 13 10 1
  
  5 7 12
```

Question 17  (1 marks)
Give the python code for finding a parent of an index in a max-heap, where the underlying array starts at 0 (rather than at 1 as in the textbook).

Answer:

```
def parent(i):
    return (i-1)//2
```

Question 18  (1 marks)
Explain what mutable versus immutable means in python. Give an example of each.

Answer:

```
# Immutable
a = [1, 2, 3, 4, 5]
a[0] = 10
print(a)  # [10, 2, 3, 4, 5]

# Mutable
b = [1, 2, 3, 4, 5]
b[0] = 10
c = b
print(c)  # [10, 2, 3, 4, 5]
```

Question 19  (1 marks)
extend is a method for list objects that concatenates another list onto its end. What values do \( a \), \( b \), and \( c \) have after this is executed. Why might you argue that this is not a good example of programming style. How would you change it?

```python
def foo(d,e):
    d.extend(e)
    return d

a = [1,2,3,4,5]
b = [6,7,8]
c = foo(a,b)
```

Answer:
Question 20  (1 marks)
Here is the code for the huffman algorithm. Show how the huffman code is built for the following alphabet and its frequency.

b  c  d  f  g  h
10  13  18  2  4  5

HUFFMAN(C)
1  n = |C|
2  Q = C
3  for i = 1 to n - 1
4     allocate a new node z
5     z.left = x = EXTRACT-MIN(Q)
6     z.right = y = EXTRACT-MIN(Q)
7     z.freq = x.freq + y.freq
8     INSERT(Q, z)
9  return EXTRACT-MIN(Q)  // return the root of the tree

Answer: Join f and g. (f,g) has freq 6
Join h and (f,g). (h,(f,g)) has freq 11
Join b and (h,(f,g)). (b,(h,(f,g))) has freq 21
Join c and d. (c,d) has freq 31
Join (b,(h,(f,g))) and (c,d). ((b,(h,(f,g))),(c,d)) has freq 52

Question 21  (1 marks)
Explain how this exhibits optimal substructure.

Answer:

Question 22  (1 marks)
Explain how this algorithm has the greedy property.

Answer: