C-10.1 Provide an example instance of the fractional knapsack problem where a greedy strategy based on repeatedly choosing as much of the highest-benefit item as possible results in a suboptimal solution.

Sol. With a knapsack of weight 10, consider items with weight-benefit pairs, (1, 40) and (10, 50). This greedy choice picks all of item 2, with benefit 50, whereas choosing item 1 first, and 9 units of item 2, gives benefit 40 + 45 = 95.

A-10.1 In the art gallery guarding problem we are given a line L that represents a long hallway in an art gallery. We are also given a set $X = \{x_0, x_1, \ldots, x_{n-1}\}$ of real numbers that specify the positions of paintings in this hallway. Suppose that a single guard can protect all the paintings within distance at most 1 of his or her position (on both sides). Design an algorithm for finding a placement of guards that uses the minimum number of guards to guard all the paintings with positions in $X$.

Sol. We can use a greedy algorithm, which seeks to cover all the designated points on $L$ with the fewest number of length-2 intervals (for such an interval is the distance one guard can protect). This greedy algorithm starts with $x_0+1$ and covers all the points that are within $[x_0, x_0+2]$. If $x_i$ is the next uncovered point, then we repeat this same covering step starting from $x_i + 1$. We then repeat this process until we have covered all the points in $X$.

R-11.1 Characterize each of the following recurrence equations using the master theorem (assuming that $T(n) = c$ for $n < d$, for constants $c > 0$ and $d \geq 1$).

a) $T(n) = 2T(n/2) + \log n$

b) $T(n) = 8T(n/2) + n^2$

c) $T(n) = 16T(n/2) + (n \log n)^4$

d) $T(n) = 7T(n/3) + n$

e) $T(n) = 9T(n/3) + n^3 \log n$

Sol.

a) $a=b=2$, $f(n) = \log n$: $T(n)$ is $O(n)$, case 1 of Master Theorem.

b) $a=8$, $b=2$, $f(n) = n^2$: $T(n)$ is $O(n^3)$, case 1 of Master Theorem.

c) $a=16$, $b=2$, $f(n) = (n \log n)^4$: $T(n)$ is $O(n^4 \log^2 n)$, case 3 of Master Theorem.

d) $a=7$, $b=3$, $f(n) = n$: $T(n)$ is $O(n^{\log_3 7})$, case 1 of Master Theorem.

e) $a=7$, $b=3$, $f(n) = n^3 \log n$: $T(n)$ is $O(n^3 \log n)$, case 3 of Master Theorem.

R-11.4 A complex number $a + bi$, where $i = \sqrt{-1}$, can be represented by the pair $(a,b)$. Describe a method performing only three real-number multiplications to compute the pair $(e,f)$ representing the product of $a + bi$ and $c + di$.

Sol. For complex factors $(a,b)$ and $(c,d)$ define the following factors

\[
\alpha = ac \\
\beta = bd
\]

\[
e = \alpha + \beta \\
f = \alpha \beta
dataculation.
\[ \gamma = (a+b)(c+d) \]

Now, we can write \((a+bi)(c+di)= (\alpha - \beta, \gamma - \alpha - \beta)\)

**C-11.3** There is a sorting algorithm, “Stooge-sort,” which is named after the comedy team, “The Three Stooges.” If the input size, \(n\), is 1 or 2, then the algorithm sorts the input immediately. Otherwise, it recursively sorts the first \(2n/3\) elements, then the last \(2n/3\) elements, and then the first \(2n/3\) elements again. The details are shown in Algorithm 11.5. Show that Stooge-sort is correct and characterize the running time, \(T(n)\), for Stooge-sort, using a recurrence equation, and use the master theorem to determine an asymptotic bound for \(T(n)\).

Algorithm StoogeSort(A, i, j):

*Input:* An array, \(A\), and two indices, \(i\) and \(j\), such that \(1 \leq i \leq j \leq n\)

*Output:* Subarray, \(A[i..j]\), sorted in non-decreasing order

\[
\begin{align*}
\text{n} & \leftarrow j - i + 1 \quad \text{// The size of the subarray we are sorting} \\
\text{if } n = 2 & \text{ then} \\
& \text{if } A[i] > A[j] \text{ then} \\
& \quad \text{Swap } A[i] \text{ and } A[j] \\
\text{else if } n > 2 & \text{ then} \\
& \quad m \leftarrow n/3 \\
& \quad \text{StoogeSort}(A, i, j - m) \quad \text{// Sort the first part} \\
& \quad \text{StoogeSort}(A, i + m, j) \quad \text{// Sort the last part} \\
& \quad \text{StoogeSort}(A, i, j - m) \quad \text{// Sort the first part again}
\end{align*}
\]

return \(A\)

**Algorithm 11.5:** Stooge-Sort

**Sol.** The proof of correctness is a simple induction proof and is omitted here. The recurrence for \(T(n)\) is \(T(n) = 3T(2n/3)+bn\), or \(T(n) = 3T(2n/3)+b\) (if we can do constant-time array passing), which, by the master theorem, is \(O(n\log 3/ \log(3/2))\), which is roughly \(O(n^{2.7})\)

**R-12.1** What is the best way to multiply a chain of matrices with dimensions that are \(10\times5\), \(5\times2\), \(2\times20\), \(20\times12\), \(12\times4\), and \(4\times60\)? Show your work.

**Sol.**

\(N_{ij} \rightarrow \text{minimum number of multiplication}\)

<table>
<thead>
<tr>
<th>i/j</th>
<th>(j=1)</th>
<th>(j=2)</th>
<th>(j=3)</th>
<th>(j=4)</th>
<th>(j=5)</th>
<th>(j=6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i=1</td>
<td>0</td>
<td>100</td>
<td>500</td>
<td>820</td>
<td>756</td>
<td>2356</td>
</tr>
<tr>
<td>i=2</td>
<td>0</td>
<td>200</td>
<td>600</td>
<td>616</td>
<td>1656</td>
<td></td>
</tr>
<tr>
<td>i=3</td>
<td>0</td>
<td>0</td>
<td>480</td>
<td>576</td>
<td>1056</td>
<td></td>
</tr>
<tr>
<td>i=4</td>
<td>0</td>
<td>0</td>
<td>960</td>
<td>5760</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i=5</td>
<td>0</td>
<td>0</td>
<td>2880</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i=6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
According to table and backtracking the optimal grouping is 
(10 x 5.5 x 2)((2 x 20.20 x 12).12 x 4).4 x 60) 
Minimum number of multiplication is 2,356

**R-12.2** Design an efficient algorithm for the matrix chain multiplication problem that outputs fully a parenthesized expression for how to multiply the matrices in chain using the minimum number of operations

**Sol** Below algorithm produces parenthesized output of the solution to the matrix chain multiplication problem. The basic idea is to use a hash map to store the grouping that produced the minimum work at each step of the calculation, then once the final result is calculated, recursively substitute the parenthesized sub problems, producing the result.

```plaintext
Algorithm 2 Produces the parenthesized output of the best matrix chain multiplication approach.

1: procedure PARENTHESEIZE(A)  > A is a list of matrices
2:     result ← []
3:     lookup ← HashMap()
4:     for i = 0 to A.length do
5:         N_{i, i} ← 0
6:     end for
7:     for i = 1 to A.length do
8:         for j = 0 to A.length - i - 1 do
9:             l ← i + j
10:            N_{i, l} ← +∞
11:                for k = i to l - 1 do
12:                   N_{i, l} ← min\{N_{j, k}, N_{j, k} + N_{k+1, l} + d_jd_{k+1}d_{l+1}\}
13:                    lookup.add((i, l), GetParenthesesString(i, j, k, l))
14:                end for
15:         end for
16:     end for
17:     minRepresentation ← RecursiveReplacement(minRepresentation, lookup)
18: end procedure

19: procedure RecursiveReplacement(representation, lookup)
20:     if representation.length = 2 then
21:         return lookup(representation)
22:     else if representation.split.length = 3 then
23:         return RecursiveReplacement(representation.split[0], lookup) +
24:             RecursiveReplacement(representation.split[1], lookup)
25:             RecursiveReplacement(representation.split[2], lookup)
26:     else
27:         return RecursiveReplacement(representation.split[0], lookup) +
28:             RecursiveReplacement(representation.split[1], lookup)
29:     end if
30: end procedure
```

**R-12.5** Let $S = \{a, b, c, d, e, f, g\}$ be a collection of objects with benefit-weight values, $a: (12,4)$, $b: (10,6)$, $c: (8,5)$, $d: (11,7)$, $e: (14,3)$, $f: (7,1)$, $g: (9,6)$. What is an optimal solution to the 0-1 knapsack problem for S assuming we have a sack that can hold objects with total weight 18? Show your work.
Please have a look at Page 345 of the book. Follow the algorithm given in that. With total weight <= 18, we get the following:

\[
\begin{align*}
B[0] &= 0 \\
B[1] &= 7 \\
B[2] &= 7 \\
B[3] &= 14 \\
B[7] &= 26 \\
B[8] &= 33 \\
B[9] &= 33 \\
B[10] &= 33 \\
B[12] &= 34 \\
B[13] &= 41 \\
B[14] &= 43 \\
B[15] &= 44 \\
B[16] &= 44 \\
B[17] &= 44 \\
B[18] &= 44
\end{align*}
\]

Optimal solution is a,d,e,f.

**R-12.6** Suppose we are given a set of telescope observation requests, specified by triples of (s,f,b) defining the start times, finish times, and benefits of each observation request as \( L = \{(1,2,5),(1,3,4),(2,4,7),(3,5,2),(1,6,3),(4,7,5),(6,8,7),(7,9,4)\} \). Solve the telescope scheduling problem for this set of observation requests.

**Sol** Note that the observation requests in this problem is already sorted by non-decreasing finish times. Denote \( B[i] \) as the maximum benefit that can be achieved with the first \( I \) requests in \( L \).

Then we have \( B[0] = 0, B[i] = \max \{B[i-1], B[pred(i)] + b[i]\} \), where \( pred(i) \) denotes the largest index \( j < I \) such that the requests \( I \) and \( j \) don’t conflict and if there is no such index, define it to be 0.

Hence we can easily know that:

\[
\begin{align*}
pred(1) &= 0 \\
pred(2) &= 0 \\
pred(3) &= 1 \\
pred(4) &= 2 \\
pred(5) &= 1 \\
pred(6) &= 3 \\
pred(7) &= 5 \\
pred(8) &= 5
\end{align*}
\]

R-12.8 Show the longest common subsequence table, L, for the following two strings:

X = “skullandbones”

Y = “lullabybabies”

What is a longest common subsequence between these strings?

Sol Please find below table L. This table is drawn by using the algorithm given on page 341 of the book

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>k</th>
<th>u</th>
<th>l</th>
<th>l</th>
<th>a</th>
<th>n</th>
<th>d</th>
<th>b</th>
<th>o</th>
<th>n</th>
<th>e</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>l</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>u</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>l</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>l</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>i</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>s</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Result of L

The longest common subsequence of “skullandbones” and “lullabybabies” is “ullabes”, of length 7.