1. 1-1

<table>
<thead>
<tr>
<th></th>
<th>1 Second</th>
<th>1 minute</th>
<th>1 Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lg n$</td>
<td>$2^{(10^6)}$</td>
<td>$2^{(6 \times 10^7)}$</td>
<td>$2^{(3.6 \times 10^9)}$</td>
</tr>
<tr>
<td>$\sqrt{n}$</td>
<td>$10^{12}$</td>
<td>$3.6 \times 10^{15}$</td>
<td>$1.3 \times 10^{19}$</td>
</tr>
<tr>
<td>$n$</td>
<td>$10^6$</td>
<td>$6 \times 10^7$</td>
<td>$3.6 \times 10^9$</td>
</tr>
<tr>
<td>$n \lg n$</td>
<td>$6.3 \times 10^4$</td>
<td>$2.8 \times 10^6$</td>
<td>$1.3 \times 10^8$</td>
</tr>
<tr>
<td>$n^2$</td>
<td>$10^4$</td>
<td>$7.7 \times 10^4$</td>
<td>$6 \times 10^4$</td>
</tr>
<tr>
<td>$n^3$</td>
<td>$10^7$</td>
<td>$10^{(10^6)}$</td>
<td>$10^{(3 \times 10^9)}$</td>
</tr>
<tr>
<td>$2^n$</td>
<td>20</td>
<td>26</td>
<td>31</td>
</tr>
<tr>
<td>$n!$</td>
<td>9</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

2. 2.2-3. Average number $A(n)$ of elements to check in order to find an element in the array of length $n$ can be obtained as:

$$A(n) = \sum_{i=1}^{n} P_i T_i$$

Where $P_i$ is probability of the element to be in position $i$, and $T_i$ is number of comparisons to make for a element in position $i$.

If we assume that an array contains distinct elements, and the element to be searched is equally likely to be in any position in the array, then $P_i = \frac{1}{n}$ and $T_i = i$. This yields

$$A(n) = \frac{1}{n} \sum_{i=1}^{n} \frac{n(n+1)}{2} = \frac{n+1}{2}$$

Thus, on average, about half the array will be checked. Running time for this algorithm is $\frac{n+1}{2} = \Theta(n)$. In the worst case, the whole array has to be checked, i.e. $n$ comparisons are to be made. Worst case running time is $n = \Theta(n)$.

3. 2.3-1. Merge sort of the array $A = \{3, 41, 52, 26, 38, 57, 9, 49\}$

Initial Array

<table>
<thead>
<tr>
<th>3</th>
<th>41</th>
<th>52</th>
<th>26</th>
<th>38</th>
<th>57</th>
<th>9</th>
<th>49</th>
</tr>
</thead>
</table>

Step 1

| 3 | 41 | 26 | 52 | 38 | 57 | 9 | 49 |

Step 2

| 3 | 26 | 41 | 52 | 9 | 38 | 49 | 57 |

Sorted Array

| 3 | 9 | 26 | 38 | 41 | 49 | 52 | 57 |

4. 3.1-4.

Is $2^{n+1} = O\left(2^n\right)$? **YES**

Is $2^{2n} = O\left(2^n\right)$? **NO** $2^{2n} = O\left(4^n\right)$
5. 3.2-2. Prove $a^\log_b n = n^\log_b a$

\[
\log_b a \cdot \log_b n = \log_b n \cdot \log_b a \quad \text{commutativity of} \quad \cdot \\
\log_b a^{\log_b n} = \log_b n^{\log_b a} \quad \text{(log } x) y = \log x^y \quad \text{(both sides)} \\
\log_b n^\log_b a = \log_b a^{\log_b n} \quad \text{(both sides)}
\]

6. 3-2.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>O</th>
<th>o</th>
<th>Ω</th>
<th>ω</th>
<th>Θ</th>
</tr>
</thead>
<tbody>
<tr>
<td>lg^n</td>
<td>n</td>
<td>YES</td>
<td>YES</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>c^n</td>
<td>n</td>
<td>YES</td>
<td>YES</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>√n</td>
<td>n^\sin n</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2^n</td>
<td>2^{n^2}</td>
<td>No</td>
<td>No</td>
<td>YES</td>
<td>YES</td>
<td>No</td>
</tr>
<tr>
<td>n^{\lg c}</td>
<td>c^{\lg n}</td>
<td>YES</td>
<td>No</td>
<td>YES</td>
<td>No</td>
<td>YES</td>
</tr>
<tr>
<td>lg(n!)</td>
<td>lg(n^n)</td>
<td>YES</td>
<td>No</td>
<td>YES</td>
<td>No</td>
<td>YES</td>
</tr>
</tbody>
</table>

7. 3-4.

a.) $f(n) = O(g(n))$ implies $g(n) = O(f(n))$ **FALSE**

Let $f(n) = n$ and $g(n) = n^2$.

$n^2$ is an upper bound on $n$, but $n$ is not an upper bound on $n^2$.

b.) $f(n) + g(n) = \Theta(\min(f(n), g(n)))$. **FALSE**

Let $f(n) = n$, $g(n) = n^2$. So for $f(n) + g(n)$, $\min(f(n), g(n)) = n$.

Thus, $n + n^2 = \Theta(n)$ does not hold.

c.) $f(n) = O(g(n))$ implies $\lg(f(n)) = O(\lg(g(n)))$, where $\lg(g(n)) \geq 0$ and $f(n) \geq 1$ for all sufficiently large $n$. **TRUE**

$f(n) = O(g(n))$ means $\exists n_0, c_1$ such that $0 \leq f(n) \leq c_1 g(n)$ for all $n > n_0 \quad (1)$

Taking $\lg$ gives $0 \leq \lg(f(n)) \leq \lg c_1 + \lg(g(n))$ for all $n > n_0 \quad (2)$

For sufficiently large $n$ in (2), $0 \leq \lg(f(n)) \leq \lg c_1 + \lg(g(n)) \leq 2\lg(g(n))$. 
d.) \( f(n) = O(g(n)) \) implies \( 2f(n) = O(2^g(n)) \) \textbf{FALSE}

Let \( f(n) = 2n \) and \( g(n) = n \). \( 2^{2n} = 4^n \notin O(2^n) \)

e.) \( f(n) = O(f(n)^2) \) \textbf{FALSE}

Let \( f(n) \) be \( \frac{1}{n} \).

f.) \( f(n) = O(g(n)) \) implies \( g(n) = \Omega(f(n)) \) \textbf{TRUE}

By transpose symmetry, \( f(n) = O(g(n)) \) iff \( g(n) = \Omega(f(n)) \)

g.) \( f(n) = \Theta\left(\frac{n}{2}\right) \) \textbf{FALSE}

Let \( f(n) = 2^n \). \( 2^n \notin O\left(\frac{n}{2}\right) \)

h.) \( f(n) + o(f(n)) = \Theta(f(n)) \) \textbf{TRUE}

For whatever \( g(n) = o(f(n)) \) is chosen, there exist \( c_0, c_1 \) such that \( g(n) \leq cf(n) \) when \( n > n_0 \).
From this and exercise 3.1 -1 (in Notes 2), \( \max(g(n), cf(n)) = cf(n) = \Theta(f(n)) \).

8. A.1-1. Simplify the expression \( \sum_{k=1}^{n} (2k - 1) \)

\[ \sum_{k=1}^{n} (2k - 1) = \sum_{k=1}^{n} 2k - \sum_{k=1}^{n} 1 = 2 \frac{n(n+1)}{2} - n = n^2 \]

9. A.2-2. Find an asymptotic upper bound on \( \sum_{k=0}^{\lfloor \log n \rfloor} \left\lfloor \frac{n}{2^k} \right\rfloor \).

\[ \sum_{k=0}^{\lfloor \log n \rfloor} \left\lfloor \frac{n}{2^k} \right\rfloor \leq \sum_{k=0}^{\lfloor \log n \rfloor} \left( \frac{n}{2^k} + 1 \right) = \log n + 1 + n \sum_{k=0}^{\lfloor \log n \rfloor} \frac{1}{2^k} \leq \log n + 1 + n \left( \frac{1}{1 - \frac{1}{2}} \right) = \log n + 1 + 2n = O(n) \]
10. A.2-3. Show that the \( n \)th harmonic number is \( \Omega(\lg n) \) by splitting summations.

\[
H_n = \sum_{k=1}^{n} \frac{1}{k} \geq \sum_{i=0}^{\lfloor \lg n \rfloor - 1} \sum_{j=0}^{2^i} \frac{1}{2^i + j} \\
= \sum_{i=0}^{\lfloor \lg n \rfloor - 1} \sum_{j=0}^{2^i} \frac{1}{2^i + j} \\
= \frac{1}{2} \sum_{i=0}^{\lfloor \lg n \rfloor - 1} 2^i \frac{1}{2^i + 1} = \frac{1}{2} \sum_{i=0}^{\lfloor \lg n \rfloor - 1} 2^i = \frac{1}{2} \sum_{i=0}^{\lfloor \lg n \rfloor} 2^i = \frac{1}{2} \lfloor \lg n \rfloor
\]

11. A.2-4. Approximate \( \sum_{k=1}^{n} k^3 \) with an integral.

\[
\int_{0}^{n} k^3 \, dk \leq \sum_{k=1}^{n} k^3 \leq \int_{0}^{n} k^3 \, dk \\
\left[ \frac{k^4}{4} \right]_{0}^{n} \leq \sum_{k=1}^{n} k^3 \leq \left[ \frac{k^4}{4} \right]_{0}^{n+1} \\
\frac{n^4}{4} \leq \sum_{k=1}^{n} k^3 \leq \frac{(n+1)^4}{4} - \frac{1}{4}
\]

12. 4.3-2. Show that the solution of \( T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + 1 \) is \( O(\lg n) \). (Assume base 2 logarithms.)

Must show that \( T(n) \leq c \lg n \) for some \( c > 0 \).

Assume \( T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) \leq c \lg \left(\frac{n}{2} \right) \) and substitute.

\[
T(n) \leq c \lg \left(\frac{n}{2} \right) + 1 = c \lg n - c + 1 \leq c \lg n \text{ if } c \geq 1
\]

13. 4.3-3. Show that the solution of \( T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + n \) is \( \Omega(n \lg n) \) and conclude that the solution is \( \Theta(n \lg n) \).

Must show that \( T(n) \geq cn \lg n \) for some \( c > 0 \).

Assume that \( T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) \geq c \frac{n}{2} \lg \frac{n}{2} \). (Base 2 logarithms)

\[
T(n) \geq 2c \frac{n}{2} \lg \frac{n}{2} + n \\
= cn \lg n - cn + n \\
\geq cn \lg n \text{ if } 0 < c \leq 1
\]
14. Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 3T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + n$. Use the substitution method to verify your answer.

![Recursion Tree](image)

Substitution method: Show $T(n) = O\left(n^{\log_2 3}\right)$

Assume $T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) \leq c\left(\frac{n}{2}\right)^{\log_2 3} - c\frac{n}{2}$. Show $T(n) \leq cn^{\log_2 3} - cn$.

$T(n) \leq 3\left(c\left(\frac{n}{2}\right)^{\log_2 3} - c\frac{n}{2}\right) + n = 3cn^{\log_2 3} - 3c\frac{n}{2} + n$

$= cn^{\log_2 3} - cn + \frac{1}{2}cn + n$

$\leq cn^{\log_2 3} - cn$ if $c \geq 2$
15. 4.4-6. Argue that the solution to the recurrence $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$ is $\Omega(n \log n)$ by appealing to the recursion tree.

From Figure 4.6, the least depth of complete levels is $\log_3 \frac{n}{2}$, and each level adds $n$ to the algorithm’s running time.

16. C.3-1. Expectation of the sum

<table>
<thead>
<tr>
<th>Sum</th>
<th>Number of ways we can get by throwing the dice</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>42</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>12</td>
</tr>
</tbody>
</table>

Total 252

Since the probability of all events is equal, $P = 1/36$. The expectation = $252 / 36 = 7$

**Expectation of the maximum**

<table>
<thead>
<tr>
<th>Maximum</th>
<th>Number of ways we can get by throwing the dice</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>66</td>
</tr>
</tbody>
</table>

Total 161

Since the probability of all events is equal, $P = 1/36$. The expectation = $161 / 36 = 4.47$

17. C.3-2

The expectation of the index of the maximum element in the array A is,

$$
\text{Expectation} = \sum_{i=1}^{n} \frac{i}{n} = \frac{1}{n} \sum_{i=1}^{n} i = \frac{1}{n} \frac{n(n+1)}{2} = \frac{n+1}{2}
$$
The probability of the maximum element in any of the n positions is \(\frac{1}{n}\). Similarly E for the minimum element is also \(\frac{n+1}{2}\).

18. C.3-3. There are four possible outcomes,

1. The person loses a dollar or
2. He gains 1 dollar or
3. He gains 2 dollars or
4. He gains 3 dollars

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Probability</th>
<th>Value Lost/Gained</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>((5/6 \times 5/6 \times 5/6))</td>
<td>-1</td>
</tr>
<tr>
<td>+1</td>
<td>((1/6 \times 5/6 \times 5/6))*3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>((1/6 \times 1/6 \times 5/6))*3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>((1/6 \times 1/6 \times 1/6))</td>
<td>3</td>
</tr>
</tbody>
</table>

Expectation = \(-\frac{125}{216} + \frac{75}{216} + \frac{30}{216} + \frac{3}{216}\)  
= \(-\frac{17}{216}\)

19. 6.2-5. Iterative Heapify

Heapify (A, i)
{
    do
    {
        p = i
        l = left(i)
        r = right(i)

        if (l <= heapsize(A) and A[l] > A[i])
            then
                largest = l
            else
                largest = i

        if (r <= heapsize(A) and A[r] > A[largest])
            then
                largest = r

        if (largest != i)
            then
                Exchange(A[i], A[largest])
                i = largest
    }
    while (p != i)
}
20. 6.4-1.

Initial

5
17 13
20 25
8 4

Build-Max-Heap

25
13 20
8 17 7
5 4

Remove 25 and Heapify

20
13 17
8 7 4
5 25

Remove 20 and Heapify

5 25
13 20 8 17 7 4

Remove 17 and Heapify

8
13 20 25
5 4

Remove 13 and Heapify

7
13 17 8 4 13 20 25

Remove 8 and Heapify

5 25
7
17 8 13 17 20 25

Remove 7 and Heapify

5 25
4 20 25

Remove 5 and Heapify

4 25
2 8 13 17 7 8 13 17 20 25

Remove 4 and Heapify

2 5 4 25
7 8 13 17 7 8 13 17 20 25
21. 6.5-1.
Initial

Remove root
Replace with 1

Swap 1 with larger child

Swap 1 with larger child

Swap 1 with larger child

22. 6.5-2.
Initial

Attach $-\infty$

Increase to 10 and ripple
23. 15.2-1
Finding the optimal parenthesization of a matrix-chain product whose sequence of dimensions is <5, 10, 3, 12, 5, 50, 6>.

The m table
\[
\begin{array}{cccccc}
0 & 150 & 330 & 405 & 1655 & 2010 \\
0 & 0 & 360 & 330 & 2430 & 1950 \\
0 & 0 & 0 & 180 & 930 & 1770 \\
0 & 0 & 0 & 0 & 3000 & 1860 \\
0 & 0 & 0 & 0 & 0 & 1500 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

The s table
\[
\begin{array}{cccccc}
0 & 1 & 2 & 2 & 4 & 2 \\
0 & 0 & 2 & 2 & 2 & 2 \\
0 & 0 & 0 & 3 & 4 & 4 \\
0 & 0 & 0 & 0 & 4 & 4 \\
0 & 0 & 0 & 0 & 0 & 5 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

The optimal parenthesization
\[(A_1 * A_2) * ((A_3 * A_4) * (A_5 * A_6))\]

24. 15.2-2
MATRX-CHAIN-MULTIPLY (A, s, i, j)
{
    if i = j
        C ← A_i
    else
        A ← MATRX-CHAIN-MULTIPLY (A, s, i, s[i, j])
        B ← MATRX-CHAIN-MULTIPLY (A, s, s[i,j]+1, j)
        C ← MATRX-MULTIPLY (A, B)
        return (C)
}

25. 15.2-5
We can find the sum by making a note of the access pattern for the m table.

Example: For n = 5

\[
\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
1 & 4 & 3 & 2 & 1 & 0 \\
2 & X & 4 & 3 & 2 & 1 \\
3 & X & X & 4 & 3 & 2 \\
\end{array}
\]
$X$ – Don’t Care; The numbers in the cells indicate the number of accesses to the cell.

It can be seen that

$$R(i,j) = \sum_{i=1}^{n} \sum_{j=1}^{n} i(i-1)$$

$$= \sum_{i=1}^{n} i(i^2 - 1)$$

$$= \frac{n}{6} ((n+1)(2n+1)) - \frac{n}{2} (n+1)$$

$$= \frac{n^3 - n}{3}$$

26. 15.4-1

$$x = <1,0,0,1,0,1,0,1,0> \quad y = <0,1,0,1,1,0,1,1,0>$$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_j$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i$</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>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>
27. 15.4-2
Print LCS without the b table

```c
print_lcs (i,j) {
    int i,j; /* i, j are the lengths of the two lists */
    if ((i == 0) || (j == 0)) return;
    if(x[i] == y[j]) /* x, y are the two lists whose LCS is to be found */
    {
        print_lcs (i-1, j-1);
        printf("%d", x[i]);
    }
    else if (c[i-1][j] >= c[i][j]) /* c is the c table as in the algorithm */
    {
        print_lcs (i-1, j);
    }
    else
    {
        print_lcs (i, j-1);
    }
    return;
}
```

28. 15.4-5
Algorithm
- a[1..n] is the input sequence
- length[1..n] contains the length of the monotonically increasing subsequences up to a[i] = { i = 1..n}
- lms is the length of the longest monotonically increasing subsequence

for i = 2 to n do
    Begin
        for j = 1 to i-1 do
            Begin
                Search for the j such that length[j] is the largest and a[i] can be included in the subsequence it represents.
            End
            length[i] = length[j] + 1
            if lms < length[i] then lms = length[i]
    End

29. 15-3
Bitonic TSP

Points P0… Pn-1 are sorted by increasing X-Coordinate
C(i, j) = Cost of achieving optimal pair of paths such that are paths ends with Pi, the other with Pj ( i < j)

Base Case
C(0,1) = dist (0, 1)

General Case
C(i-1, i) = min {C(j, i-1) + dist(j, i)}
$0 \leq j \leq i-1$

$C(i, j) = C(i, j-1) + \text{dist}(j-1, j)$ where $i < j-1$

Final solution

$$\min \{ C(i, n-1) + \text{dist}(i, n-1) \}$$

$0 \leq i < n-1$

**30. 16.1-1**

/* $f[1..n]$ contains finishing times (sorted) of activities

$s[1..n]$ contains the starting times of those activities

$m[1..n]$ contains the number of activities from 1 .. $i$ that can be scheduled $m_i$ in the

problem

$fm[1..n]$ indicates the finishing times of the tasks scheduled in each of $m[1..n]$ */

Begin

$m[1] = 1$

$fm[1] = f[1]$

for $i = 2$ to $n$

Begin

if ($fm[i-1] \leq s[i]$) then

Begin

$m[i] = m[i-1] + 1$

$fm[i] = f[i]$

End

else

Begin

$fm[i] = fm[i-1]$

$m[i] = m[i-1]$

End

End i

End

**31. 16.1-4**

$n \leftarrow \text{length}[s]$

for $i \leftarrow 1$ to $n$

$A[i] \leftarrow \{ \emptyset \}$ //each $A[i]$ (lecture Hall) has a set of activities

LIST_INSERT($L, i$);

$k \leftarrow 0$

while $L \neq \emptyset$

Do $k \leftarrow k + 1$

$i \leftarrow \text{head}[L]$

for $j \leftarrow i + 1$ to $\text{tail}[L]$

Do if $s_j \leq f_i$

then $A[k] \leftarrow A[k] \cup \{ j \}$

$i \leftarrow j$

LIST_DELETE($L, j$)

return $L$ // the final value of ‘k’ has the number of lecture halls
32. 16.2-4
The greedy strategy would be to fill up the water bottle at the last moment i.e., Travel to the farthest water station that can be reached from the current water station (without falling short)

33. 16.2-5
Sort the points in ascending order of their k values
The greedy strategy would be to enclose the leftmost unenclosed point and all points that lie within a unit distance of this point. The next interval will begin at the closest point to the right of this interval

34. 16.3-3

Generalization:
code = \begin{cases} 
    k-1 \text{ } 1\text{'s followed by a '0', if } k \leq n-1 \\
    k \text{ } 1\text{'s, } k = n
\end{cases}