4. i) let Array A[1...n] is input array of numbers.

let Array L[1...n] stores length of LIS, where L[i] indicates the length of LIS ending at index i such that A[i] is included as last element of LIS.

$$L[i] = \begin{cases} 1 & \text{if } i=1 \\ 1 + \max_{1 \le j < i} (L[j]) & \text{if } A[j] < A[i] \\ 1 & \text{if no } j \text{ exists where } A[j] < A[i] \end{cases}$$

LIS of Array may end with any number in the Array.

∴ LIS will be maximum value of Array L

$$LIS = \max_{1 \le i \le n} [L(i)]$$

2) Algorithm

LIS(A[1...n], n)

1. let L[1...n] // stores length of LIS ending at each index
2. L[i] = 1
3. for i = 2 to n
4. L[i] = 1
5. for j = 1 to i-1
6. if A[i] > A[j]
7. L[i] = max(L[i], L[j]+1)

// Both for loops end here

// return max value of Array L

8. max_value = L[1]
9. for i = 2 to n
10. max_value = max(max_value, L[1])
11. return max_value.

Time Complexity:-

Line 3 - n times

Line 4 - n-1 times

Line 5 to 7 -

for i=2, 1 time
i=3, 2 times
⋮
i=n, n-1 times

Line 9 - n times

Line 10 - n-1 times

$$T(n) = c_1 n + \sum_{i=1}^{n-1} i$$

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

$$T(n) = \theta(n^2)$$

3. Chess board - n x m

c[i,j] - coin value at cell (i,j)

1) let P[i,j] represents maximum coins collected when robot wanders from cell (i,j) to cell (n,m)

P[1,1] will give maximum coins collected on entire chess board when wanders from cell c(1,1) to c(n,m).

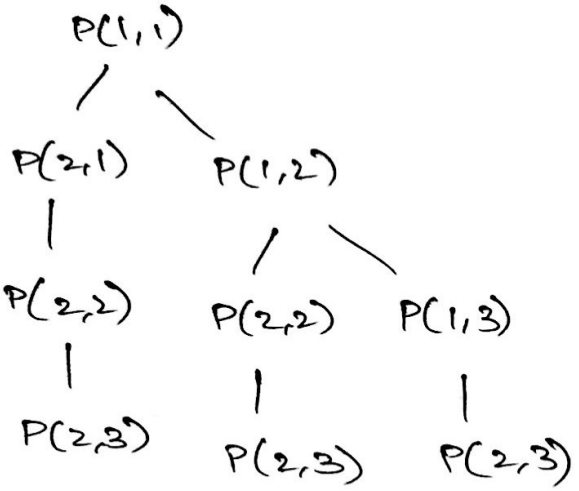
$$P[i,j] = \begin{cases} c[i,j] + \max(P[i+1,j], P[i,j+1]) & \text{if } i<n \text{ and } j<m \\ c[i,j] & \text{if } i=n \text{ and and } j=m \\ c[i,j] + P[i+1,j] & \text{if } j=m \text{ and } i<n \\ c[i,j] + P[i,j+1] & \text{if } i=n \text{ and } j<m \end{cases}$$

Pseudocode - recursively without memoization

```
1. Max-Coins(C, i, j)
2. if (i=n and j=m)
3.     return c[i,j]
4. else if (i=n)
5.     return c[i,j] + Max-Coins(C, i, j+1)
6. else if (j=m)
7.     return c[i,j] + Max-Coins(C, i+1, j)
8. else
9.     return c[i,j] + max(Max-Coins(C, i, j+1),
                           Max-Coins(C, i+1, j))
```

Time complexity

Suppose $n=2$, $m=3$



Time complexity = $2^{\max(n,m)}$

2) iterative Algorithm with memoization

```
1.  Max-coins_Memoization( c[1...n, 1...m], n, m)
2.  let P[1..n, 1..m] and b[1...n, 1...m] be new tables
3.  for i = n downto 1
4.      for j = m downto 1
5.          if(i=n and j=m)
6.              P[i,j] = c[i,j]
7.          else if (i=n)
8.              P[i,j] = c[i,j] + P[i,j+1]
9.              b[i,j] = "→"
10          else if (j=m)
11.             P[i,j] = c[i,j] + P[i+1,j]
12.             b[i,j] = "↓"
13.         else
14.             if P[i,j+1] > P[i+1,j]
15.                 P[i,j] = c[i,j] + P[i,j+1]
16.                 b[i,j] = "→"
17.             else
18.                 P[i,j] = c[i,j] + P[i+1,j]
19.                 b[i,j] = "↓"
20. return P and b.
```

P[1,1] gives maximum coins collected by robot.

Array b gives path.

```
1.  Print-Path ( b, n, m)
2.  i = 1
3.  j = 1
4.  Print i, j      // for printing cell c(1, 1)
5.  while ( i < n  or  j < m)
6.      if b[i, j] = "→"
7.          Print i, j+1
8.          j = j+1
9.      else if b[i, j] = "↓"
10.         Print i+1, j
11.         i = i+1
12. Print i, j      // for printing cell c(n, m)
```

Time-Complexity.

Each element in array should be visited once.

$\therefore T(n) = n \times m$