Solution 11

1.

		n	o	t	i	c	e
	0	0	0	0	0	0	0
e	0	0 ↑	0 ↑	0 ↑	0 ↑	0 ↑	1 ↖
n	0	1 ↖	1 ←	1 ←	1 ←	1 ←	1 ↑
c	0	1 ↑	1 ↑	1 ↑	1 ↑	2 ↖	2 ←
i	0	1 ↑	1 ↑	1 ↑	2 ↖	2 ↑	2 ↑
r	0	1 ↑	1 ↑	1 ↑	2 ↑	2 ↑	2 ↑
c	0	1 ↑	1 ↑	1 ↑	2 ↑	3 ↖	3 ←
l	0	1 ↑	1 ↑	1 ↑	2 ↑	3 ↑	3 ↑
e	0	1 ↑	1 ↑	1 ↑	2 ↑	3 ↑	4 ↖

LCS is “nice”!

2.

```
MEMOIZED-LCS-LENGTH(X, Y, i, j)
    if c[i, j] > -1
        return c[i, j]
    if i == 0 or j == 0
        return c[i, j] = 0
    if x[i] == y[j]
        return c[i, j] = LCS-LENGTH(X, Y, i - 1, j - 1) + 1
    return c[i, j] = max(LCS-LENGTH(X, Y, i - 1, j), LCS-LENGTH(X, Y, i, j - 1)
```

If your answer in problem 1 or 2 can show how your algorithm works, you can get points, or you will lose 3 points

3.

Solution by Qing Zhang

key	begin	else	end	for	if	return	while
probability	6%	10%	6%	24%	15%	30%	22%

```
table e:
    0.00    0.06    0.22    0.34    0.80    1.10    1.85    2.45
            0.00    0.10    0.22    0.62    0.92    1.67    2.21
                    0.00    0.06    0.36    0.66    1.41    1.85
                            0.00    0.24    0.54    1.23    1.67
                                    0.00    0.15    0.60    1.04
                                            0.00    0.30    0.74
                                                    0.00    0.22
                                                            0.00
table w:
    0.00    0.06    0.16    0.22    0.46    0.61    0.91    1.13
            0.00    0.10    0.16    0.40    0.55    0.85    1.07
                    0.00    0.06    0.30    0.45    0.75    0.97
                            0.00    0.24    0.39    0.69    0.91
                                    0.00    0.15    0.45    0.67
                                            0.00    0.30    0.52
                                                    0.00    0.22
                                                            0.00
table root:
      1       2       2       4       4       4       6
              2       2       4       4       4       6
                      3       4       4       4       6
                              4       4       5       6
                                      5       6       6
                                              6       6
                                                      7
```

4.

No, the algorithm does not work.

Counter-example: Take 3 nodes [A(0.32) B(0.33) C(0.35)]

The Tree according to Prof Thi's algorithm:

C

B

A

Cost: 0.35*1 + 0.33*2 + 0.32*3 = 1.97

Optimal Tree:

B

A C

Cost: 0.33*1 + 0.35*2 + 0.32*2 = 1.69.

5.

Solution by Fan Bu

We define that i is equal to the number of months passed, and j is equal to the total number of machines we have already produced (at the time after i months but before $i+1$ months). Then the cost function is as follows.

$$C(i,j) = \begin{cases} 0, & i = 0 \wedge j = 0 \\ \infty, & i = 0 \\ \min_{0 \le k \le j}\{C(i-1, j-k) + p(k) + h(i,j)\}, & others \end{cases}$$

And,

$$p(k) = \begin{cases} 0, & k \le m \\ c_1 \cdot (k-m), & k > m \end{cases}$$

And,

$$h(i,j) = \begin{cases} \infty, & j - \sum_{k=1}^{i} d_k < 0 \\ c_2 \cdot (j - \sum_{k=1}^{i} d_k), & others \end{cases}$$

Then, the answer of the problem is $C(12, D)$, where $D = \sum_{k=1}^{12} d_i$.
Here is an example. Suppose $m = 2$, and $c_1 = 2$, and $c_2 = 1$, and $d = [1,2,4,3,1,2,1,2,1,1,1,6]$.
Then, the minimum cost is 13, and the production plan is $[2,2,3,3,1,2,1,2,1,2,2,4]$.

6.

$$dp[i,j] = \begin{cases} dp[i-1][j-1] + M(a,b) & if\ X[i] = Y[i] \\ M(a,b) + min\{dp[i-1][j-1] + S(a,b),\ dp[i][j-1] + I(a),\ dp[i][j-1] + D(b) & otherwise\} \end{cases}$$

#of subproblems is m*n, running time is O(mn)

7.

$P[i] = c_i$, $m_i \leq 500$
$P[i] = \min\{ (P[j] + dcheck(i, j) + c_i) : 1 \leq j < i\}$ $m_i > 500$

$dcheck(i, j) = 0$, $m_i - m_j \leq 500$
$dcheck(i, j) = (m_i - m_j - 500)/2$, $m_i - m_j > 500$

P[i] denotes the minimum total cost when we go from the starting point and stop at the i^{th} hotel. c_i denote the cost of the i^{th} hotel. m_i denote the distance between the start and hotel i.

Optimal substructure:

Suppose that p_i is one of the optimal stops $p_1, p_2, \ldots, p_m$ we made along the route. Then $p_1, p_2, \ldots, p_{i-1}$ and $p_{i+1}, p_{i+2}, \ldots, p_m$ must also be the optimal stops for the sub problem they belong to; otherwise we simply use the cut-and-paste argument to substitute a better solution for the current solution to the sub problem, leading to a better solution to the original problem, which is a contradiction.

The minimum cost is $353. We need to stop at the 1st, 3rd, 4th, 5th, 6th, and 8th hotel.

8.

We can solve this problem by using a Greedy strategy.

In order to get home in as few days as possible, we will try to maximize the distance x we travel every day where x <= 300 due to the constraint.

```
findMinimumDays(m)
    dist = n, count = 1
    for (i = n − 1 to 1)
        if (m[dist] − m[i] > 300)
            if (m [i + 1] − m[i] > 300)  // impossible
                return -1
            dist = i + 1
            count ++
    return count
```

The time complexity of this algorithm is O(n), where n is the number of hotels.

You can also solve this problem using a Dynamic Programming approach.

9.

maxDiff[i][j] = max(values[i] - maxDiff[i + 1][j], values[j] - maxDIff[i][j - 1]) if i != j

values[i] if i == j

```java
class Solution {
    public boolean PredictTheWinner(int[] values) {
        if (values == null || values.length == 0) {
            return false;
        }
        // maxDiff contains the maximum difference to the player's opponent when he is facing to the subarray from i to j.
        int[][] maxDiff = new int[values.length][values.length];
        // choice contains the player's choice when he is facing to the subarray from i to j.
        // 0 means he chooses the first one, 1 means he chooses the last one.
        int[][] choice = new int[values.length][values.length];
        for (int i = 0; i < values.length; i++) {
            maxDiff[i][i] = values[i];
        }
        for (int length = 2; length <= values.length; length++) {
            for (int i = 0; i <= values.length - length; i++) {
                int j = i + length - 1;
                int diff1 = values[i] - maxDiff[i + 1][j];
                int diff2 = values[j] - maxDiff[i][j - 1];
                if (diff1 >= diff2) {
                    maxDiff[i][j] = diff1;
                    choice[i][j] = 0;
                } else {
                    maxDiff[i][j] = diff2;
                    choice[i][j] = 1;
                }
            }
        }
        // Compose the result.
        List<Integer> result = new ArrayList<>(values.length);
        int i = 0;
        int j = values.length - 1;
        while (i <= j) {
            if (choice[i][j] == 0) {
                result.add(i);
                i++;
            } else {
                result.add(j);
                j--;
            }
        }
        System.out.println(result);
        // Return if the first player can win.
        return maxDiff[0][values.length - 1] >= 0;
    }
}
```

Related questions:

https://leetcode.com/problems/predict-the-winner/

https://leetcode.com/problems/stone-game/