Dynamic Programming Problems

These are the sample problems which are intended to provide a hands-on introduction to solving problems using Dynamic Programming.

1. Coin Change Problem
2. Optimum Path Problem
3. Billboard Location Problem
4. Additional Problem - DNA Sequence Alignment

Problem 1 - Coin Change Problem

Objective: Given a amount A and n coins, $c_1 < c_2 < c_3 < \ldots < c_n$write a program to calculate the minimum number of coins required to make the exact change for the amount 'A'.

Bonus Points: Use backtracking to identify which set of coins have been used to solve the problem.

Test Cases:
Use these sample test cases to ensure that your solution returns the correct result.

Coins	Target Value	Expected Min Coins
1,2,5,10,20,50	1	1 → {1}
1,2,5,10,20,50	3	2 → {2, 1}
1,2,5,10,20,50	5	1 → {5}
1,2,5,10,20,50	9	3 → {5, 2, 2}
1,2,5,10,20,50	11	2 → {10, 1}
1,2,5,10,20,50	23	3 → {20, 2, 1}
1,2,5,10,20,50	49	5 → {20, 20, 5, 2, 2}
1,2,5,10,20,50	87	5 → {50, 20, 10, 5, 2}
1,4,15,20,50	3	3 → {1, 1, 1}
1,4,15,20,50	10	4 → {4, 4, 1, 1}
1,4,15,20,50	23	3 → {15, 4, 4}
1,4,15,20,50	32	3 → {15, 15, 1, 1}

Problem 2 - Optimum Path Through Matrix

Objective: Given a 2-D matrix of positive integers, find the path from the top-left corner to the bottom-right of the matrix where the elements on the path have the smallest total sum. You are only allowed to move either right or down (i.e. can't move up, to the left. or diagonally)

This is a sample grid with the optimum path highlighted in blue.

8	7	4	2	3	5
3	2	1	2	5	2
2	4	5	3	3	6
6	4	5	4	3	2
1	3	4	5	6	3
7	3	2	1	3	5

$Path_{opt} = 8 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 3 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 5 = 35$

Hint: There are only 2 ways to get to the final square. Which option should be selected?

Bonus Points:
As well as providing the minimum total cost to reach the bottom right corner, also provide the path that was used to achieve this.

Test Cases:
Case 1:

8	**7**	**4**	**2**	**3**	**5**
3	2	1	2	5	2
2	4	5	3	3	6
6	4	5	4	3	2
1	3	4	5	6	3
7	3	2	1	3	5

$Score = 35$
$Path_{opt} = 8 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 3 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 5 = 35$

Case 2:

2	**5**	**17**	**13**	**1**	**11**	**10**	**6**	**19**	**20**	**10**	**17**	**14**	**15**	**11**	**2**	**4**	**8**	**13**	**11**
10	13	11	14	3	12	14	16	9	14	2	20	12	4	12	8	3	6	5	4
8	8	12	10	3	6	1	3	9	10	6	3	16	3	4	8	11	16	19	8
13	14	6	18	11	17	5	18	9	20	4	12	18	16	15	19	11	1	17	10
8	16	10	1	3	9	3	14	5	8	8	17	7	13	14	3	18	4	5	4
8	15	18	7	17	7	10	20	2	14	2	15	8	9	8	19	8	1	20	17
12	6	11	14	17	15	14	7	5	19	4	15	20	14	14	6	4	8	8	14
10	8	17	2	3	4	17	19	17	18	12	16	1	1	6	10	14	16	5	4
15	20	4	16	12	7	13	3	19	9	15	12	8	20	4	20	10	13	20	16
20	19	10	7	17	9	1	18	16	19	20	2	9	1	7	8	20	19	18	19

$Score = 201$
$Path_{opt} = 2 \rightarrow 5 \rightarrow 17 \rightarrow 13 \rightarrow 1 \rightarrow 3 \rightarrow 3 \rightarrow 6 \rightarrow 1 \rightarrow 3 \rightarrow 9 \rightarrow 10 \rightarrow 6 \rightarrow 3 \rightarrow 16 \rightarrow 3 \rightarrow 4 \rightarrow 8 \rightarrow 11 \rightarrow 11 \rightarrow 1 \rightarrow 4 \rightarrow 1 \rightarrow 8 \rightarrow$

Problem 3 - Billboard Locations

Objective: Suppose you're managing construction of billboards on the M62 between Leeds & Manchester, a distance of M miles. The possible sites for advertising hoardings are given by numbers $x_1 < x_2 < \cdots < x_n$, each in the interval $[0, M]$, specifying their position in miles measured from Leeds. If you place a billboard at position x_i , you receive a revenue of $r_i > 0$.

Regulations imposed by the Department of Transport require that no two billboards be within n miles or less of each other (i.e. the distance between successive billboards must be at least $n + 1$ miles). You'd like to place billboards at a set of the sites so as to maximize your total revenue, subject to this restriction.

Calculate the maximum achievable revenue and the location of the billboards which generates this figure.

Test Cases:
Case 1
Locations : $x = [6, 7, 12, 13, 14]$
Revenues : $r = [5, 6, 5, 3, 1]$
Total Distance : $M = 20$
Gap : $n = 5$
Expected Answer
Max Revenue = 10
Placements = $[6, 12]$

Case 2
Locations : $x = [6, 7, 12, 13, 14]$
Revenues : $r = [5, 6, 5, 3, 1]$
Total Distance : $M = 20$
Gap : $n = 4$
Expected Answer
Max Revenue = 11
Placements = $[7, 12]$

Case 3
Locations : $x = [6, 7, 10, 12, 13, 14, 17, 19, 22, 23, 26, 28, 29, 30]$
Revenues : $r = [4, 11, 8, 5, 3, 5, 9, 5, 4, 5, 3, 3, 4, 7]$
Total Distance : $M = 30$
Gap : $n = 4$
Expected Answer
Max Revenue = 37
Placements = $[7, 12, 17, 23, 30]$

Case 4
Locations : $x = [1, 3, 6, 7, 10, 12, 13, 14, 17, 19, 22, 23, 26, 28, 29, 30]$
Revenues : $r = [3, 6, 4, 11, 8, 5, 3, 5, 9, 5, 4, 5, 3, 3, 4, 7]$
Total Distance : $M = 30$
Gap : $n = 5$
Expected Answer
Max Revenue = 35
Placements = $[3, 10, 17, 23, 30]$

Additional Problem - DNA Sequence Alignment

We want to find the best way to align 2 sequences of DNA to minimise the number of differences between corresponding characters. For each, character in the sequences you have the option of either using the given character or inserting a gap "— " which will allow the subsequent characters to align better.

For example, if we aligned the following 2 sequences **ACGGTC** and **ATCGCC** we could produce an optimal alignment as:

```
A-CGGTC
ATCG-CC
```

This would have an 'Edit Distance' of 3 since there are 3 positions (2, 5 & 6) where the characters in the 2 generated sequences are different. This distance is called the Levenshtein Distance and forms the basis of the Needleman-Wunsch Algorithm which determines the distance between 2 DNA or Protein sequences in bioinformatics.

This problem should build upon your solution to the Optimum Path problem with the following difference that allows an additional option to move diagonally Left & Down in a single step. It uses an insertion cost of 1 for each gap inserted and a substitution cost of one if a diagonal move is made when the 2 characters are different.

🏆 **Bonus Points**:
As well as calculating the Edit Distance, generate the pair of sequences containing the gaps which produce the best possible score.

🎯 **Test Cases**:
Seq1 : **ACGGTC**
Seq2 : **ATCGCC**

Edit Distance = 3
Optimal Alignments = **A-CGGTC** , **ATCG-CC** (other alignments may also be valid)