Tutorial Sections 5-6

1. You are managing a small gardening business that specialises in cultivating a rare type of plant. These plants can be cut and sold in segments, and each segment's length can vary. The unique aspect of these plants is that the price of each segment depends not just on its length but also on demand for specific lengths. Your task is to determine how to cut a plant of a given length to maximise your total revenue.

   The selling prices for different lengths of plant segments are as follows:

Length (units)	Price (£)
1	2
2	4
3	7
4	3
5	9

   (a) Let $P(n)$ denote the maximum price achievable by cutting a plant of length n. Which recursion problem should you solve to ascertain the optimal strategy for cutting the plant?

   (b) What would be the optimal cutting length if you had a plant that is 5 units long if you want to maximise revenue?

   (c) How many different calculations you would need to perform to solve this problem using a fully exhaustive approach? How does that compare to the above strategy?

Solution: This is a modified version of the rod-cutting problem presented in the reading material (**CLRS**, section 14.1).

(a) To solve the problem, we need to determine the optimal way to cut a 5-unit long plant to maximise the following recursion formula:

$$P(n) = \max_{1 \le i \le n} \{Price(i) + P(n-i)\},$$

with $P(0) = 0$ as the base case.

(b) Using a bottom-up approach, for $n = 1$ (length 1 unit), we have:

$$P(1) = \max\{Price(1)\} = \max\{2\} = 2$$

For $n = 2$,

$$P(2) = \max\{Price(1) + P(1), Price(2)\} = \max\{2 + 2, 4\} = 4$$

For $n = 3$,

$$P(3) = \max\{Price(1) + P(2), Price(2) + P(1), Price(3)\} = \max\{2 + 4, 4 + 2, 7\} = 7$$

For $n = 4$,

$$\begin{aligned} P(4) = \max\{&Price(1) + P(3), Price(2) + P(2), Price(3) + P(1), Price(4)\} \\ &= \max\{2 + 7, 4 + 4, 7 + 2, 3\} = 9 \end{aligned}$$

Finally, for $n = 5$ we have:

$$\begin{aligned} P(5) = \max\{&Price(1) + P(4), Price(2) + P(3), Price(3) + P(2), Price(4) + P(1), Price(5)\} \\ &= \max\{2 + 3, 4 + 7, 7 + 4, 3 + 2, 9\} = 11 \end{aligned}$$

Therefore, to achieve the maximum revenue of £11 from a 5-unit plant, the optimal strategy involves cutting the plant into lengths that correspond to the calculation of $P(5) = 11$. From the calculation, we can see that this configuration corresponds to 2 units + 3 units.

(c) To solve the problem using a fully exhaustive approach, we would need to calculate the revenue for every possible combination of cuts that can be made on a 5-unit long plant. Considering all possible partitions of 5, that would give us a total of 16 distinct ways (which corresponds to $2^{n-1} = 2^{5-1} = 2^4$, since they have an independent option of cutting, or not cutting, at length i units from the one end).
DP (Bellman's recursion) reduces the number of calculations to 5 (the number of calculations is linear with respect to the length of the plant). This may not be that different for such a simple problem, but as N increases DP significantly reduces the number of calculations.

2. Imagine that you are a logistics coordinator of an online store planning the route for a delivery truck. The truck needs to make deliveries to three different locations in a city. Your task is to plan the most efficient route that minimises the total distance travelled until the last city. The distances (in miles) between the locations (including the starting point) are as follows:

	Starting Point	Location 1	Location 2	Location 3
Starting Point	-	10	20	30
Location 1	10	-	25	15
Location 2	20	25	-	10
Location 3	30	15	10	-

(a) Explain how recursion can be used to find the shortest path considering all previous steps, and use this approach to find the shortest path for the truck.

(b) Consider the following recursion to find the minimum path:

$$C(S, i) = \min_{j \in S, j \neq i} (C(S \backslash \{i\}, j) + d_{j,i}),$$

which represents the minimum cost of a path that starts at the starting point, visits each location in set S, and ends at location i. How this strategy can be used to simplify the problem?

Solution: This is essentially the same travelling salesman problem from the previous tutorial.

(a) We can use recursion to systemically explore all permutations of routes and calculate their total distances. The recursion formula $C(S, i)$ represents the minimum cost of a path that starts at the starting point, visits each location in set S, and ends at location i. By recursively solving smaller sub-problems and using their solutions to construct solutions for larger problems (larger subsets), we can find the shortest path efficiency. For this, we can define the base case as the starting point (in which the cost is 0) and then find subsets of size 1 (when S contains only one location). For larger subsets, we can recursively call the function for smaller subsets of locations and add the distance from the last location to the current one. At each stage n, we can extend the optimal configuration X^*_{n-1} in the previous stage to obtain the current optimal configuration X^*_n, as defined by Equation (5.3) in the notes.

(b) We can use Bellman recursion to solve this problem. For the base case, we have $C(\{SP\}, SP) = 0$; for the subsets of size 1 and 2 there will be only one possible term to consider in the set S. For subsets of size 3, we could simplify the number of calculations based on the previous

(smaller, simpler) subproblems.
For subsets of size 1, we can calculate the length of the path that starts at the starting point, visits a location j and ends at the location j:

$$P_{SP\to L1} = P_{SP\to SP} + d_{SP,L1} = 0 + 10 = 10$$
$$P_{SP\to L2} = P_{SP\to SP} + d_{SP,L2} = 0 + 20 = 20$$
$$P_{SP\to L3} = P_{SP\to SP} + d_{SP,L3} = 0 + 30 = 30$$

The corresponding optimal configuration corresponds to $X_1^* = [SP, L1]$, which can be extended in the next iteration.
For subsets of size 2, i.e., starts at the SP, visits city i and city j, ending at location j, we have:

$$P_{SP\to L1\to L2} = P_{SP\to L1} + d_{L1\to L2} = 10 + 25 = 35$$
$$P_{SP\to L1\to L3} = P_{SP\to L1} + d_{L1\to L3} = 10 + 15 = 25,$$

which gives an optimal configuration $X_2^* = [SP, L1, L3]$. For subsets of size 3, we could simplify the number of calculations based on the previous (smaller, simpler) subproblems:

$$P_{SP\to L1\to L3\to L2} = P_{SP\to L1\to L3} + d_{L3\to L2} = 25 + 10 = 35$$

From all the calculations, we can see that the shortest path is SP $\to$ L1 $\to$ L3 $\to$ L2, for a total distance of 35 miles.

3. You are helping to organise a multi-cultural food festival and need to select a diverse menu featuring four different dishes from various cuisines. The selection of dishes is guided by their popularity ratings, determined through surveys of the local community prior to the event. This year, the popularity ratings for each dish are as follows:

Food	Local Popularity
Pizza	4.2
Sushi	3.9
Curry	4.4
Tacos	3.7
Pad Thai	4.3
Dim Sum	3.8
Crepes	4.1
Falafel	4.0

(a) Last year, the festival featured falafel, tacos, sushi, and dim sum on its menu. Starting with the menu from last year, your goal is to create a menu with the highest combined popularity rating. Due to the festival's theme of cultural diversity, you must ensure a wide representation of cuisines. You can only swap out one dish at a time to maintain a balance in the menu.

(b) How much more effective would the algorithm be if using a 2-Hamming neighbour? (Note: you can answer this question with reference to the size of the full configuration space without going through any computation.)

Solution: This is a contextualised problem of the maximum sum combination problem from the notes.

(a) We can apply the greedy approximate search algorithm to find the optimal choice of the menu by maximising the popularity:

$$F(X) = \sum_{i=1}^{4} x_i,$$

where x_i represents the popularity of the i-th dish in the configuration X. Our problem is then to find the optimal configuration of dishes,

$$X^* = \arg\max_{X' \in \mathcal{X}} F(X').$$

- At $n = 0$, our initial configuration is given by $X_0 = \{4.0, 3.7, 3.9, 3.8\}$, for a combined popularity of $F(X_0) = 15.4$.
- The *neighbourhood search* consists in finding one dish among all new dishes that can be swapped in the current configuration. Performing this search, we see that the highest objective function value is the swap of tacos (3.9) by the curry (4.4), so the updated optimal configuration will be $X_1 = X' = \{4.0, \mathbf{4.4}, 3.9, 3.8\}$, with a combined popularity of $F(X') = 16.1 > F(X_0)$; therefore, we continue the iteration;
- A new *neighbourhood search* yields the highest cost function by swapping the Dim Sum (3.8) by the Pad Thai (4.3), for a total popularity of $F(X') = 16.6 > F(X_1)$. The updated configuration will then be $X_2 = \{4.0, 4.4, 3.9, \mathbf{4.3}\}$;
- In the next iteration, we would swap the Sushi (3.9) by the Pizza (4.2), resulting in $F(X') = 16.9 > F(X_2)$ and $X_3 = \{4.0, 4.4, \mathbf{4.2}, 4.3\}$;
- Similarly, we swap the Falafel (4.0) by the Crepes (4.1) in the next iteration, resulting in $F(X') = 17.0 > F(X_3)$ and $X_4 = \{\mathbf{4.1}, 4.4, 4.2, 4.3\}$;

- Since X_4 contains the four highest scores in its configuration, the next neighbourhood search would yield $F(X') \leq F(X_4)$ for any candidate configuration, which means that the algorithm terminates with $X^* = X_4$.

Therefore, the optimal menu for this year will be crepes, curry, pizza, and pad thai, which produces a combined popularity of 17.

(b) The 2-Hamming neighbour means that we would change 2 elements of the configuration in each step (rather than swapping one at a time). This would be more effective because we would reach the optimal solution in just two iterations (for this specific problem). On the other hand, the number of configurations to explore in the search increases (it grows combinatorially with the distance).