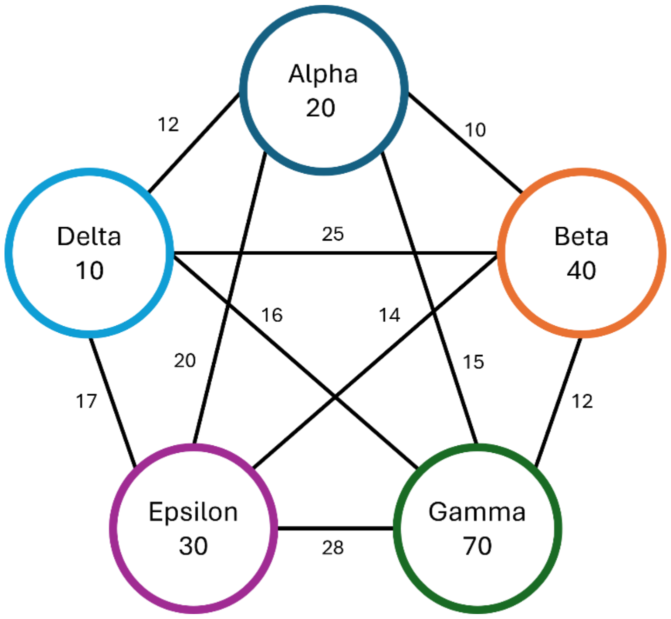
Computational Methods Assignment

Task 4

Dynamic Programming is a way of designing algorithms where you divide the problem into subproblems, find solutions to those subproblems and put them back together to form a complete solution to the problem we want to solve. (*Dynamic Programming W3Schools)*



To help us with this is a key part of dynamic programming and a technique used to solve problems, called tabulation. Tabulation uses a table consisting of the results of the most basic subproblems, then gets filled with more and more subproblem results until we find the result to the complete problem. (*DSA Tabulation W3Schools)*

Once again, we're going to start off on Delta since we know through brute forcing that this is the fastest starting planet. We begin by creating a table of all 4 possible routes arriving from Delta. To help make our table easy to read I'm going to order my values from least to most expensive, alongside leaving the multiplication of each number by 25 until the end.

D A (120)	D G (160)	D E (170)	D B (250)

Now we have our initial subproblem solved, we can work up the "tree" to work out the next step of subproblems, which in our case are the next planets in the route.

In the below table, you can see all the possible routes consisting of 3 planets originating from Delta. I've also included my working out to show my formula.

D A (120)	D G (160)	D E (170)	D B (250)
D A B (420)	D G B (1120)	D E B (730)	D B A (750)
D A G (570)	D G A (1360)	D E A (970)	D B G (850)
D A E (720)	D G E (2400)	D E G (1290)	D B E (950)

10 * 12	10 * 16	10 * 17	10 * 25
120 + (10 + 20) * 10	160 + (10 + 70) * 12	170 + (10 + 30) * 14	250 + (10 + 40) * 10
120 + (10 + 20) * 15	160 + (10 + 70) * 15	170 + (10 + 30) * 20	250 + (10 + 40) * 12
120 + (10 + 20) * 20	160 + (10 + 70) * 28	170 + (10 + 30) * 28	250 + (10 + 40) * 14

Now we have this above table, we need to eliminate all of the duplicate combinations that we don't need, for each combination we only want to proceed with it if it's the cheapest. In a coded example this would save unnecessary processing costs by calculating the combinations for expensive routes.

D A (120)	D G (160)	D E (170)	D B (250)
D A B (420)	D G B (1120)	D E B (730)	D B A (750)
D A G (570)	D G A (1360)	D E A (970)	D B G (850)
D A E (720)	D G E (2400)	D E G (1290)	D B E (950)

`

D A (120)	D E (170)	D B (250)
D A B (420)	D E B (730)	D B G (850)
D A G (570)	D E G (1290)	
D A E (720)		

After we simplify our table to only have cheap routes, we can continue to calculate our next subproblem, which is a route that is 4 planets long.

D A (120)	D E (170)	D B (250)
D A B (420)	D E B (730)	D B G (850)
D A G (570)	D E G (1290)	
D A E (720)		
D A B G (1260)	D E B A (1530)	D B G A (2650)
D A G B (1770)	D E G A (2940)	
D A E B (1560)		
D A B E (1400)	D E B G (1690)	D B G E (4210)
D A G E (3370)	D E G B (2610)	
D A E G (2400)		

10 * 12	10 * 17	10 * 25
120 + (10 + 20) * 10	170 + (10 + 30) * 14	250 + (10 + 40) * 12
120 + (10 + 20) * 15	170 + (10 + 30) * 28	
120 + (10 + 20) * 20		
420 + (10 + 20 + 40) * 12	730 + (10 + 30 + 40) * 10	850 + (10 + 40 + 70) * 15

570 + (10 + 20 + 70) * 12	1290 + (10 + 30 + 70) * 15	
720 + (10 + 20 + 30) * 14		
420 + (10 + 20 + 40) * 14	730 + (10 + 30 + 40) * 12	850 + (10 + 40 + 70) * 28
570 + (10 + 20 + 70) * 28	1290 + (10 + 30 + 70) * 12	
720 + (10 + 20 + 30) * 28		

Like before, we need to eliminate all of the redundant combinations we don't need and simplify the table.

D A (120)	D E (170)	D B (250)
D A B (420)	D E B (730)	D B G (850)
D A G (570)	D E G (1290)	
D A E (720)		
D A B G (1260)	D E B A (1530)	D B G A (2650)
D A G B (1770)	D E G A (2940)	
D A E B (1560)		
D A B E (1400)	D E B G (1690)	D B G E (4210)
D A G E (3370)	D E G B (2610)	
D A E G (2400)		

D A (120)	D E (170)
D A B (420)	D E B (730)
D A G (570)	D E G (1290)
D A E (720)	
D A B G (1260)	D E B G (1690)
D A B E (1400)	
D A E G (2400)	

Lastly, we perform our last subproblem which is the complete 5 planet route.

D A (120)	D E (170)
D A B (420)	D E B (730)
D A G (570)	D E G (1290)
D A E (720)	
D A B G (1260)	D E B G (1690)
D A B E (1400)	
D A E G (2400)	
D A B G E (5180)	D E B G A (3940)
D A B E G (4200)	
D A E G B (3960)	

10 * 12	10 * 17
120 + (10 + 20) * 10	170 + (10 + 30) * 14
120 + (10 + 20) * 15	170 + (10 + 30) * 28
120 + (10 + 20) * 20	

420 + (10 + 20 + 40) * 12	730 + (10 + 30 + 40) * 10
570 + (10 + 20 + 70) * 12	1290 + (10 + 30 + 70) * 15
720 + (10 + 20 + 30) * 14	
1260 + (10 + 20 + 40 + 70) * 28	1690 + (10 + 30 + 40 + 70) * 15
1400 + (10 + 20 + 40 + 30) * 28	
2400 + (10 + 20 + 30 + 70) * 12	

Here we can see our cheapest route is Delta → Epsilon → Beta → Gamma → Alpha. Now when we multiply this by our 25 intergalactic currency for fuel, we get 98500.

Compared to other algorithms, such as
Greedy Algorithm which gave us Delta → Beta → Alpha → Epsilon → Gamma = 123,750
and Brute Forcing which gave us Delta → Alpha → Epsilon → Beta → Gamma = 69,000,
we can see that with Dynamic Programming we get close to the actual answer in a lot less time than it takes to brute force something.

The time complexity for Dynamic Programming is calculated by the number of unique subproblems multiplied by the time taken per subproblem.
To help us calculate this there are n subproblems and each subproblem is solved in a constant time. Resulting in O(n * 1) (*Demystifying Dynamic Programming, 2018)*

Bibliography

DSA Tabulation W3Schools. Available at: https://www.w3schools.com/dsa/dsa_ref_tabulation.php (Accessed: 26 November 2024).

Dynamic Programming W3Schools. Available at: https://www.w3schools.com/dsa/dsa_ref_dynamic_programming.php (Accessed: 26 November 2024).

Demystifying Dynamic Programming (2018) freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/demystifying-dynamic-programming-24fbdb831d3a/ (Accessed: 26 November 2024).