Q1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Vehicle #** | **Location name** | **Location ID** | **Latitude (y)** | **Longitude (x)** | **Cuml. Distance travelled** | **Delivery amount** |
| **1** | Depot | 0 | 50 | 50 | 0 | 0 |
| Customer 15 | 15 | 37.4 | 35.5 | 19.209633 | 597.3 |
| Customer 9 | 9 | 36.5 | 44 | 27.75714726 | 666 |
| Depot | 0 | 50 | 50 | 42.53043396 | 0 |
| **2** | Depot | 0 | 50 | 50 | 0 | 0 |
| Customer 11 | 11 | 87.5 | 7.3 | 56.82904187 | 421.2 |
| Customer 19 | 19 | 86.4 | 15.1 | 64.70622411 | 51.5 |
| Customer 2 | 2 | 76.4 | 36.4 | 88.23684246 | 709.8 |
| Customer 6 | 6 | 72.6 | 74 | 126.0283759 | 26.3 |
| Customer 16 | 16 | 54.4 | 75.3 | 144.2747454 | 667.8 |
| Depot | 0 | 50 | 50 | 169.954504 | 0 |
| **3** | Depot | 0 | 50 | 50 | 0 | 0 |
| Customer 13 | 13 | 11 | 78.9 | 48.54080757 | 389.9 |
| Customer 3 | 3 | 0.9 | 61.7 | 68.48698515 | 244.4 |
| Customer 10 | 10 | 1.8 | 60.9 | 69.69114461 | 397.6 |
| Customer 5 | 5 | 20.6 | 51.4 | 90.75509486 | 722.6 |
| Customer 14 | 14 | 23 | 47.5 | 95.33439611 | 84.2 |
| Depot | 0 | 50 | 50 | 122.4498898 | 0 |
| **4** | Depot | 0 | 50 | 50 | 0 | 0 |
| Customer 17 | 17 | 33.6 | 87.1 | 40.56316063 | 315.6 |
| Customer 12 | 12 | 13.6 | 96.5 | 62.66202938 | 848.2 |
| Customer 8 | 8 | 13.6 | 93 | 66.16202938 | 791.7 |
| Depot | 0 | 50 | 50 | 122.4999385 | 0 |
| **5** | Depot | 0 | 50 | 50 | 0 | 0 |
| Customer 4 | 4 | 42.7 | 7.3 | 43.31951062 | 33 |
| Customer 18 | 18 | 9.1 | 6.3 | 76.93438827 | 743.1 |
| Customer 7 | 7 | 5.3 | 10.4 | 82.52455822 | 801.9 |
| Customer 1 | 1 | 10.5 | 27.9 | 100.78079 | 240.1 |
| Depot | 0 | 50 | 50 | 146.0429155 | 0 |

Table 1: VRP Spreadsheet Solutions

Q2

Our proposed heuristic for the knapsack algorithm consists of three steps.

1. Create a scoring list for the points based on an equation which basically aims to get maximum benefit while minimizing the weight and volume required.
2. Sort *score* value in descending order.
3. Add the items by searching one by one from score list. If an item is eligible in terms of weight and volume capacity, add the item. Otherwise, go to the next best scored item.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input #** | **Final Profit** | **Final Weight** | **Final Volume** |
| 1 | 77641 | 673.2 | 3339.2 |
| 2 | 58076 | 772.9 | 6199.4 |
| 3 | 66148 | 894.1 | 7281.7 |
| 4 | 75340 | 713.1 | 4345.6 |
| 5 | 53068 | 541.5 | 4322.1 |
| 6 | 50761 | 418.1 | 3232.9 |
| 7 | 56595 | 461.7 | 2307.6 |
| 8 | 66825 | 553.8 | 3125.8 |
| 9 | 61876 | 554.7 | 3668.7 |
| 10 | 77594 | 760.7 | 4327.3 |
| 11 | 74957 | 720.6 | 6936.3 |
| 12 | 49766 | 346.9 | 2344.2 |

Table 2: Results of the Example Inputs

|  |  |
| --- | --- |
| Input # | Items (Sequenced in order of addition) |
| 1 | [3, 11, 10, 19, 5, 16, 20, 7, 18, 14, 13, 1, 15, 4, 17, 8] |
| 2 | [6, 4, 11, 1, 18, 5, 9, 14, 13, 20, 15, 2, 7, 8, 16, 12, 19, 10, 17, 3] |
| 3 | [13, 15, 16, 10, 20, 2, 12, 3, 7, 8, 11, 9, 19, 17, 14, 5, 1, 4, 18, 6] |
| 4 | [17, 4, 11, 10, 6, 14, 2, 1, 12, 20, 3, 8, 19, 9, 7, 15, 5, 18] |
| 5 | [13, 16, 17, 11, 12, 2, 7, 10, 3, 4, 18, 1, 5, 15, 6] |
| 6 | [12, 10, 3, 5, 13, 19, 9, 14, 2, 17, 6, 16] |
| 7 | [11, 4, 10, 5, 6, 17, 15, 20, 2, 1, 8, 7] |
| 8 | [3, 13, 7, 10, 1, 5, 8, 4, 2, 6, 9, 18, 12] |
| 9 | [7, 11, 2, 12, 18, 4, 9, 8, 16, 10, 14, 5, 17] |
| 10 | [1, 14, 13, 16, 4, 3, 9, 18, 6, 12, 19, 7, 8, 15, 20, 11, 5, 2] |
| 11 | [3, 1, 20, 17, 2, 18, 14, 7, 9, 6, 4, 8, 16, 5, 11, 19, 15, 13, 12, 10] |
| 12 | [9, 1, 17, 19, 3, 11, 2, 13, 20, 15, 14] |

Table 3: Items that are listed in a bin for Example Inputs