**Design and Analysis of Algorithm (CSC4202)**

**Second Semester, 2024/2025**

**ASSIGNMENT 2**

**INSTRUCTIONS:**

1. Submit via PutraBlast:
2. The files containing your source code. Make sure that the files compile and run.
3. The output and the complete comparison table.
4. Submission dateline is **6 June 2025**.
5. Coping or other forms of cheating is forbidden. The faculty has very strong rules about this, and the standard penalty for first offence is to award 0 marks to all parties concerned.
6. Use Java to implement your solution.

**QUESTION:**

**Real-life carry-on knapsack**

As a nomad and live out of one carry-on bag, the total weight of all his worldly possessions must fall under airline cabin baggage weight limits - usually 10kg. On some smaller airlines, however, this weight limit drops to 7kg. Occasionally, he has to decide not to bring something with him to adjust to the smaller weight limit.

As a practical exercise, deciding what to leave behind (or get rid of altogether) entails laying out all his things and choosing which ones to keep. That decision is based on the item's usefulness to him (its worth) and its weight.

Being a programmer, you are aware that decisions like this could be made more efficiently by a computer. It's done so frequently and so ubiquitously, in fact, that many will recognize this scenario as the classic *packing problem* or *knapsack problem.* How do you go about telling a computer to put as many important items in his bag as possible while coming in at or under a weight limit of 7kg? With *algorithms!*

Based on the data below with three columns: the item's name (a string), a representation of its worth (an integer), and its weight in grams (an integer). There are 40 items in total and represented worth by ranking each item from 40 to 1, with 40 being the most important.

| **Item** | **Worth** | **Weight** |
| --- | --- | --- |
| Lenovo X1 Carbon (5th Gen) | 40 | 112 |
| 10 pairs thongs | 39 | 80 |
| 5 Underarmour Strappy | 38 | 305 |
| 1 pair Uniqlo leggings | 37 | 185 |
| 2 Lululemon Cool Racerback | 36 | 174 |
| Chargers and cables in Mini Bomber Travel Kit | 35 | 665 |
| The Roost Stand | 34 | 170 |
| ThinkPad Compact Bluetooth Keyboard with trackpoint | 33 | 460 |
| Seagate Backup PlusSlim | 32 | 159 |
| 1 pair black denim shorts | 31 | 197 |
| 2 pairs Nike Pro shorts | 30 | 112 |
| 2 pairs Lululemon shorts | 29 | 184 |
| Isabella T-Strap Croc sandals | 28 | 200 |
| 2 Underarmour HeatGear CoolSwitch tank tops | 27 | 138 |
| 5 pairs black socks | 26 | 95 |
| 2 pairs Injinji Women's Run Lightweight No-Show Toe Socks | 25 | 54 |
| 1 fancy tank top | 24 | 71 |
| 1 light and stretchylong-sleeve shirt (Gap Fit) | 23 | 147 |
| Uniqlo Ultralight Down insulating jacket | 22 | 235 |
| Patagonia Torrentshell | 21 | 301 |
| Lightweight Merino Wool Buff | 20 | 50 |
| 1 LBD (H&M) | 19 | 174 |
| Field Notes Pitch Black Memo Book Dot-Graph | 18 | 68 |
| Innergie PocketCell USB-C 6000mAh power bank | 17 | 14 |
| Important papers | 16 | 228 |
| Deuter First Aid Kit Active | 15 | 144 |
| Stanley Classic Vacuum Camp Mug 16oz | 14 | 454 |
| |  | | --- | |  |   JBL Reflect Mini Bluetooth Sport Headphones | 13 | 14 |
| Anker SoundCore nano Bluetooth Speaker | 12 | 89 |
| Oakley Latch Sunglasses | 11 | 30 |
| Ray Ban Wayfarer Classic | 10 | 45 |
| Zip bag of toiletries | 9 | 236 |
| Petzl E+LITE Emergency Headlamp | 8 | 27 |
| Laptop Bag | 7 | 20 |
| Peak Design Cuff Camera Wrist Strap | 6 | 26 |
| Travelon Micro Scale | 5 | 125 |
| BlitzWolf Bluetooth Tripod/Monopod | 4 | 150 |
| Humangear GoBites Duo | 3 | 22 |
| Touchlight | 2 | 10 |
| Vapur Bottle 1L | 1 | 41 |

Implement the two common approaches called *dynamic programming and greedy algorithm* to solve the knapsack problem to help him for packing as many items as the limit allows (**Airline limit:** 4000g) while maximizing the total worth and ranking 40 items from the most important to less important.

Based on your implementation, complete the comparison table below and choose the best possible solution in solving the given problem.

|  |  |  |
| --- | --- | --- |
|  | **Greedy algorithm** | **Dynamic programming** |
| **Total weight** |  |  |
| **Total worth** |  |  |
| **Speed** |  |  |