CS 202

Design and Analysis of Algorithms

Assignment 2

[Question 1] Bounded Knapsack

We discussed the 0-1 knapsack problem during the class on week 4. In a bounded knapsack problem, there are *n* items, each item *i* (*1 ≤ i ≤ n*) is associated with a quantity q_i , besides its weight w_i and its value v_i . The objective is to maximize the total value of the items placed into a knapsack with capacity *W*, such that the total weight of the items does not exceed *W*, and the number of copies of each item in the knapsack does not exceed the quantity of that item.

Please write a program to compute the maximum total value from a bounded knapsack problem, described by each line of test inputs. Test inputs begin with a number indicating the number of lines below. For each row below, there are *n* sets of three integers separated by colons, representing the weight w_i, the value v_i and the quantity q_i of each item *i* (*1 ≤ i ≤ n*). The last integer of the line represents the weight capacity *W* of a knapsack.

There will be 7 sets of test cases with 1 mark each. The remaining 3 marks are awarded to the justification or discussion on the time complexity of your algorithm.

Sample Input

```
8
83:98:4 69:75:3 95:103:7 66:52:4 754
74:78:9 100:108:5 69:55:3 54:58:3 608
80:89:2 90:85:2 78:65:6 86:69:2 54:44:10 994
67:69:5 93:92:5 66:54:8 61:66:7 51:50:4 56:62:7 53:50:4 1102
79:86:3 100:83:6 53:60:6 55:47:9 63:68:7 98:104:5 90:88:2 88:88:3 65:65:3 95:85:3 2223
93:77:2 66:68:8 86:75:3 56:61:8 51:56:9 60:60:7 63:69:7 97:90:2 79:94:5 100:87:10
<--continue from the line above--> 80:90:7 57:64:5 50:40:2 98:107:6 83:96:2 4162
```

```
88:77:10 77:68:8 70:80:8 54:49:9 58:48:3 55:62:4 50:60:8 61:68:10 64:62:3 51:51:7
<--continue from the line above--> 81:90:8 94:80:5 4453
72:85:4 80:70:3 61:61:3 98:115:3 96:82:6 68:60:5 56:67:5 92:86:8 86:80:8 65:76:4
<--continue from the line above --> 100:90:6 62:61:9 89:79:8 63:67:4 3450
```

Sample Output

```
833
656
880
1179
2334
4557
4613
3566
```

[Question 2] Longest Common Mountain Sub-sequence

Given two sequences $A = \{a_1, a_2, \ldots, a_{|A|}\}$ and $B = \{b_1, b_2, \ldots, b_{|B|}\}$, a sequence $C = \{c_1, c_2, \ldots, c_{|C|}\}$ is called a common mountain sub-sequence of A and B, if (i) there exist two strictly increasing functions f and g (meaning $f(x_1) < f(x_2)$ if $x_1 < x_2$), such that $c_i = a_{f(i)} = b_{g(i)}$ for all $1 \le i \le |C|$; and (ii) there exists an index k, such that $c_1 < c_2 < \ldots < c_k > c_{k+1} > \ldots > c_{|C|}$, for some k in $1 \le k \le |C|$. For example, with functions $f(1) = 2$, $f(2) = 3$, $f(3) = 5$, and $g(1) = 2$, $g(2) = 5$, $g(3) = 6$, $\{c_1, c_2, c_3\}$ is identical to $\{a_2, a_3, a_5\}$ and $\{b_2, b_5, b_6\}$. Additionally, if $c_2 > c_1$ and $c_2 > c_3$, $\{c_1, c_2, c_3\}$ forms a common mountain sub-sequence of A and B.

There are many such common mountain sub-sequences between A and B, however, there is a longest sub-sequence among them. In this question, you are to write a program which computes the length of a longest common mountain sub-sequence.

Implement an algorithm in the function **`LCMS`**. Please also state and justify the time complexity of your algorithm.

Test inputs begin with the number of pairs of sequences. Each sequence is described by one line and contains a list of positive integers in hexadecimal between 1 and 2^{24}-1. The length of each sequence is denoted as n, and $1 \le n \le 5000$. For each pair of sequences, output the length of a longest common mountain sub-sequence.

Sample Input

```
5
26 224 180 48 243 166 84 139 245 141 63 29 69
131 36 180 173 127 243 94 127 155 126 200 62 239 84 105 17 50
134 71 62 22 215 206 130 135 135 19 252 87 231 113 245 233 59 139 248 100 189 24 70
<--continue from the line above--> 253 255 245 87 50 47
251 175 80 215 52 157 49 252 116 37 133 248 33 122 103 255 143 136 40 50 202 255 141
75 176 154 227 157 173 86 216 233 52 55 250 208 129 57 22 244 153 151 129 64 104 131
<--continue from the line above--> 22 253 196 239 190 175 202 60 46 79 44 241 211
73 154 155 46 168 134 224 58 130 233 248 33 150 123 208 37 66 35 74 244 67 37 123 32
<--continue from the line above--> 176 151 205 253 25 175 210 34 46 211 169
160 138 81 224 22 217 60 156 164 114 249 147 210 152 196 239 29 86 140 147 164 98
<--continue from the line above--> 179 131 75 186 245 140 238 187 101 54 253 169 198
<--continue from the line above--> 25 201 118 26 75 118 215 79 154 216 192 186 32 234
<--continue from the line above--> 76 125 83 167 250 197 224 42 230 235 42 229 210 169
<--continue from the line above--> 91 109 44 62 166 231 217
179 208 230 173 25 37 192 207 81 117 43 65 236 156 246 61 138 70 147 87 76 239 35 115
<--continue from the line above--> 132 173 153 127 60 33 153 162 247 147 108 222 221
<--continue from the line above--> 245 39 77 108 238 89 183 195 253 102 190 156 215
<--continue from the line above--> 225 119 104 146 167 202 91 59 133 216 115 242 232
<--continue from the line above--> 186 122 101 152 234 164 30 167 227 18 170 230 119
<--continue from the line above--> 22 91 98 184 52 119 62 237
49 239 27 182 116 26 228 55 208 214 131 243 130 76 99 43 149 154 150 248 96 150 19 81
<--continue from the line above--> 45 120 172 187 175 134 236 79 41 138 150 154 81 127
<--continue from the line above--> 215 206 75 43 87 115 189 47 32 67 254 255 239 199
<--continue from the line above--> 105 16 47 93 234 208 254 231 152 234 243 220 250
<--continue from the line above--> 242 135 196 254 153 128 209 205 225 178 214 32 194
<--continue from the line above--> 103 185 64 232 50 31 217 249 212 205 55 250 29 171
<--continue from the line above--> 61 179 81 209 164 220 182 254 82 190 137 25 124 94
<--continue from the line above--> 55 87 147 152 96 164 104 23 78 32 213 62 130
27 86 26 147 140 115 191 72 83 243 239 16 188 43 180 234 96 190 32 45 65 209 188 175
<--continue from the line above--> 193 150 235 206 116 254 132 106 104 184 189 225 157
<--continue from the line above--> 236 255 17 176 37 231 18 234 48 146 233 254 204 66
<--continue from the line above--> 250 197 254 129 209 197 247 60 214 51 185 90 143 66
<--continue from the line above--> 217 192 246 101 235 171 139 224 249 123 44 231 220
<--continue from the line above--> 250 162 94 242 167 66 153 68 56 96 185 250 118 55
<--continue from the line above--> 194 32 108 46 241 255
```

Sample Output

```
3
4
6
10
15
```

[Question 3] Debug minimum coin change problem

There is a serious bug in the example code of minimum coin change (limited supply) problem, as shown in **`A2Q3.py`**. Please read the code together with the comments, answer the following questions and debug the code.

1) (1 mark) Which part is wrong? Is it **`min_coin_with_plan`** or **`min_coin_num_only`**?

2) (2 marks) Which element of dynamic programming has the wrong part violated? Is it the optimal sub-structure, or the overlapping of sub-problems?

3) (2 marks) Give a counter example that fails this code.
   When the amount for coin change = ________ cents, the corresponding optimal solution should be ________ coins, but the code gives the following results:

4) (5 marks) Propose a way to rectify the bug, please highlight your proposed correction in Python code with comments.