





cs330su21-a.sg

[Dashboard](#) / [My courses](#) / [cs330su21-a.sg](#) / [Week 11 \(19 July - 25 July\)](#) / [Programming Assignment 3 - Dynamic Programming](#)

-  [Description](#)


 [Submission](#)

 [Edit](#)

 Submission view

Grade

Reviewed on Friday, 30 July 2021, 2:35 AM by Automatic grade
grade: 100.00 / 100.00

Assessment report  [-]
[\[+\]](#) **Summary of tests**

Submitted on Friday, 30 July 2021, 2:35 AM ([Download](#))
q.cpp

```

1  /*!*****
2  \file    q.cpp
3  \author  Goh Wei Zhe
4  \par     DP email: weizhe.goh@digipen.edu
5  \par     Course: CS330
6  \par     Section: A
7  \par     Programming Assignment #3
8  \date    29-07-2021
9
10 \brief   Dynamic Programming, 0-1 Knapsack, Coin Changes, Tree Tower
11 *****/
12
13 // only include the following two head files
14 #include <iostream>
15 #include <vector>
16 #include <algorithm> // max number from a vector
17 // Don't add more
18
19 #define INT_MAXI (int) ((unsigned) -1 / 2)
20 #define INT_MINI -INT_MAXI - 1
21
22 namespace CS330
23 {
24     namespace dp
25     {
26         /*!*****
27         \brief
28         Given that weight limit as a constraint and a list of the charms with
29         their weights and desirability rating, deduce the maximum possible sum of
30         ratings.
31
32         \param W
33         An array vector of weights
34
35         \param D
36         An array vector of desirable ratings
37
38         \return
39         Returns the maximum possible sum of desirability ratings.
40         *****/
41         int charm_bracelet(int M, std::vector<int> const& W,
42             std::vector<int> const& D)
43         {
44             int size = static_cast<int>(W.size());
45             std::vector<int> Bag(M + 1);
46
47             for (int i = 0; i < size; ++i)
48             {
49                 for (int j = M; j >= W[i] ; --j)
50                 {
51                     Bag[j] = std::max(Bag[j], Bag[j - W[i]] + D[i]);
52                 }
53             }
54
55             return Bag[M];
56         }
57
58         /*!*****
59         \brief
60         Find the minimum numbers of coins for a specific change value.
61
62         \param value
63         Value to find minimum of coin change for
64
65         \param denominations
66         A vector of coins denominations
67
68         \return
69         Returns the minimum number of coins required to form the change value.
70         *****/
71         int coin_changes(int value, std::vector<int> const& denominations)
72         {
73             int size = value + 1;
74
75             //Set all values to max value
76             std::vector<int> table(size, INT_MAXI);
77
78             //Init first element to 0
79             table[0] = 0;
80
81             //Iterate through all values
82             for (int i = 1; i <= value; ++i)
83             {
84                 //Iterate through all demonimations
85                 for (const auto& x : denominations)
86                 {
87                     int index = i - x;
88                     if (i >= x && (table[index] != INT_MAXI))
89                     {
90                         table[i] = std::min(table[i], table[index] + 1);
91                     }
92                 }
93             }
94
95             if (table[value] == INT_MAXI)
96                 return -1;
97
98             return table[value];
99         }
100
101         /*!*****
102         \brief
103         Find a path from root node to one of the leaf nods that has maximal total
104         weight.
105
106         \param rows
107         Number of rows
108

```

```
109  \param cols
110  Number of cols
111
112  \param weights
113  A pointer to an int array
114
115  \return
116  Returns the maximal total weight from root node to leaf node among all
117  paths.
118  *****/
119  int tree_tower(int rows, int cols, int* weights)
120  {
121      int size = rows * cols;
122
123      //Iterate from -1 row and -1 column, bottom up approach
124      for (int i = size - rows - 1; i >= 0; i--)
125      {
126          //Get maximum value between next row index value and
127          //next row + 1 index value
128          weights[i] = std::max(weights[i + cols],
129                                weights[i + cols + 1]) + weights[i];
130      }
131
132      return weights[0];
133  }
134 }
135 }
```

[VPL](#)

◀ Attendance cs330su21-a.sg
Thursday 22/07/2021 11:00am-
12:40pm

Jump to...

Attendance cs330su21-a.sg Tuesday
27/07/2021 11:00am-12:40pm ▶

You are logged in as [Wei Zhe GOH](#) ([Log out](#))
[cs330su21-a.sg](#)
[Data retention summary](#)
[Get the mobile app](#)