Deakin University

Data Structures and Algorithms

OnTrack Submission

Programming - Problem Solving

Submitted By:
Peter Stacey
pstacey
2020/08/31 14:33

Tutor:
Maksym Slavnenko

Outcome	Weight
Complexity	♦♦♦♦◊
Implement Solutions	♦♦♦◊◊
Document solutions	♦♦♦♦◊

This task involves evaluating approaches to solving a long running problem, making use of memory to minimise runtime (eg. through memoization and using cached values), in looking at ways to optimise calculations dynamically and in coding our design to pass a series of tests. This relates to all three learning outcomes.

August 31, 2020

```cs
using System;
using System.Text;
using System.Text.RegularExpressions;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;

namespace Task_6_2
{
    public class BoxOfCoins
    {
        /// <summary>
        /// Prints out the table of values used to store the results
        /// of calculations through different pathways
        /// </summary>
        /// <param name="table">2-dimensional array to print</param>
        /// <param name="n">length or height of the square 2d array</param>
        public static void PrintTable(int[,] table, int n)
        {
            for(int i = 0; i < n; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    Console.Write("{0,7} ", table[i, j]);
                }
                Console.WriteLine();
            }
        }

        /// <summary>
        /// Calculates the maximum difference possible for alternate
        /// choice of treasure chests, picked from the left or right
        /// end of a line of chest.
        /// </summary>
        /// <param name="boxes">Array or values of the coins in each chest</param>
        /// <returns>The difference between the totals based on the order
        /// of chooisng chests (+ve difference indicates better result for
        /// person chooisng first)</returns>
        /// <exception cref="System.ArgumentNullException">thrown when the
        /// array of coin values is null</exception>
        public static int Solve(int[] boxes)
        {
            if (boxes is null) throw new ArgumentNullException();
#if DEBUG
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();
#endif
            if (boxes.Length is 1) return boxes[0]; // Nothing to calculate. Alex
            ↪  gets the coins if only 1 chest
            int n = boxes.Length;

            // Create a table to store results, allowing re-use
            // if the same result occurs in a later iteration.
```

```
53              // This saves on time by caching results to be returned.
54              // (ie. implementation of memoization)
55              // SPace complexity: O(n^2)
56              int[,] table = new int[n, n];
57              int gap;      // staerting difference in positions in the table
58              int i, j;     // pointers to cells in the table and to chests in the
                ↪  array
59              int x, y, z;  // incremental sum of possible coin combinations
60
61              // Time complexity: O(n^2)
62              // This approach minimises the re-calculation of
63              // overlapping solutions.
64              for (gap = 0; gap < n; gap++) {
65                  for (i = 0, j = gap; j < n; i++, j++) {
66                      x = ((i + 2) <= j) ? table[i + 2, j] : 0;
67                      y = ((i + 1) <= (j - 1)) ? table[i + 1, j - 1] : 0;
68                      z = (i <= (j - 2)) ? table[i, j - 2] : 0;
69
70                      table[i, j] = Math.Max(boxes[i] + Math.Min(x, y),
71                                             boxes[j] + Math.Min(y, z));
72                  }
73              }
74
75              // Account for different position of final results depending on odd or
                ↪  even number of chests
76              int result = Math.Max(table[0, n - 1] - table[1, n - 1], table[0, n -
                ↪  1] - table[0, n - 2]);
77  #if DEBUG
78              stopwatch.Stop();
79              Console.WriteLine("Result: {0}", result);
80              Console.WriteLine("Duration: {0}", stopwatch.Elapsed);
81              PrintTable(table, n);
82  #endif
83              return result;
84          }
85      }
86
87  }
```