Assignment 2

Solution

# Mark Distribution

|  |  |  |
| --- | --- | --- |
| sub question | Marks |  |
| 1. | 2 | Prove that optimal substructure property exists |
| 2. | 1 | What will be the smaller sub-problems |
| 3. | 1 | Recursive definition/formula |
| 4. | 5 | dynamic programming algorithm  1 – Optimal Solution Returned  1 – Minimum Space and Time  3 – Problem Solved Correctly |
| 5. | 1 | Space and Time Complexity |

# Key Notes

**Note for pt.** 1 - Optimal substructure states – {A} ∩ {B} -> {C} where {A} and {B} is subset of {S} that is the solution. {C} is such that it is equal to {S}.   
This means that the intersection of the best solution of sub-problems yields the best global solution.  
To explain this for specific problems, you must first prove that solution can be split and the best solution from the split can yield the best solution if we did not split the problem OR disprove that sub-problems do not yield best global solution.  
Problem -> sub-problem1 (Solution), sub-problem2 (Solution) = Problem (Solution)

**Note for pt.** 2 – Define sub-problems from base case and upwards to get the best solution easier

**Note for pt.** 3 – Prefer writing mathematical notation for recursive definition rather than code using recursion

**Note for pt.** 4 – Best solution means the answer to the problem.  
For question no. 2 (Bursting balloon problem) the best solution meant the sequence in which the balloons had to be burst. The optimal value was the best answer that could be generated with this set of balloons.  
*The idea is to not only convert the recursive problem into dynamic but also understand what each value in the array of the dynamic programming solution represents.*

* Q3 optimal solution is the floor egg was dropped from and the attempts made to reach to the solution.  
  All those who had not returned optimal solution and used a run-time of nk2 had lost 2 marks, 1 for not returning optimal solution and 1 for not writing the best run-time code.  
  Best runtime was n.lg(k).  
  However, since the optimal solution required more space to store in this run-time, if optimal solution was returned in a greater run-time, marks are awarded.
* Q4 keep in mind that the optimal solution for an array that is completely filled with negative values would be the max value from the array. i.e. [-1, -4, -3, -9]. The MSS should be -1 and not 0.  
  For all questions that made use of the if condition *“if sum < 0: sum = 0; si, ei = 0;”* your final answer would yield the MSS = -9 and arr [-9] when the answer should be -1 and arr[-1].

**Note for pt.** 5 – Write the solution for your algorithms runtime including the optimal solution generating runtime.