**Lab 10: Dynamic Programming**

**Question 1:** Consider the general instance of the following well-known problem. Give change for amount n using the minimum number of coins of denominations d1<d2 < . . .<dm. For the coin denominations used in the United States, as for those used in most if not all other countries, there is a very simple and efficient algorithm discussed in the next chapter. Here, we consider a dynamic programming algorithm for the general case, assuming availability of unlimited quantities of coins for each of the m denominations d1< d2 < . . . < dm where d1 = 1. Implement the java code using the following pseudo-code and the backtrace algorithm that prints the used coin set **(70pts)**.

Table

Description automatically generated

Text, letter

Description automatically generated

**Expected Output:**

**Denominators = {1, 3, 5}**

**Amount = 9**

**Minimum number of coin:3**

**Coin set:1 3 5**

**Q2:** What are the space and time efficiency of this algorithm? Use non-recursive analysis to calculate the time efficiency of this algorithm **(30pts)**.

**Q3:** A cutting factory purchases large metal poles, cuts them into shorter shafts, and then sells them. Each slice is free of charge. The price is determined by the length of the cut. The manufacturer wishes to optimise its earnings by determining the optimal cuts. Given below are the prices that the manufacturer can produce for each length of the stick. Determine the greatest income, rn, that can be generated by slicing the metal poles into pieces and selling them using a bottom-up dynamic programming technique **(Optional: 50pts)**.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Length | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Price | 3 | 5 | 6 | 7 | 8 | 11 | 13 |

**r7 = ?**