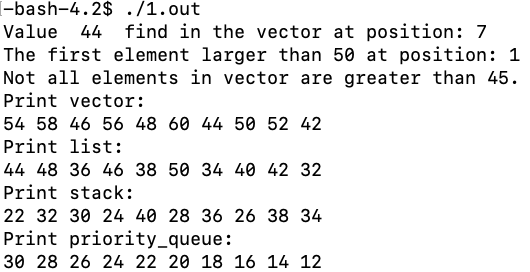
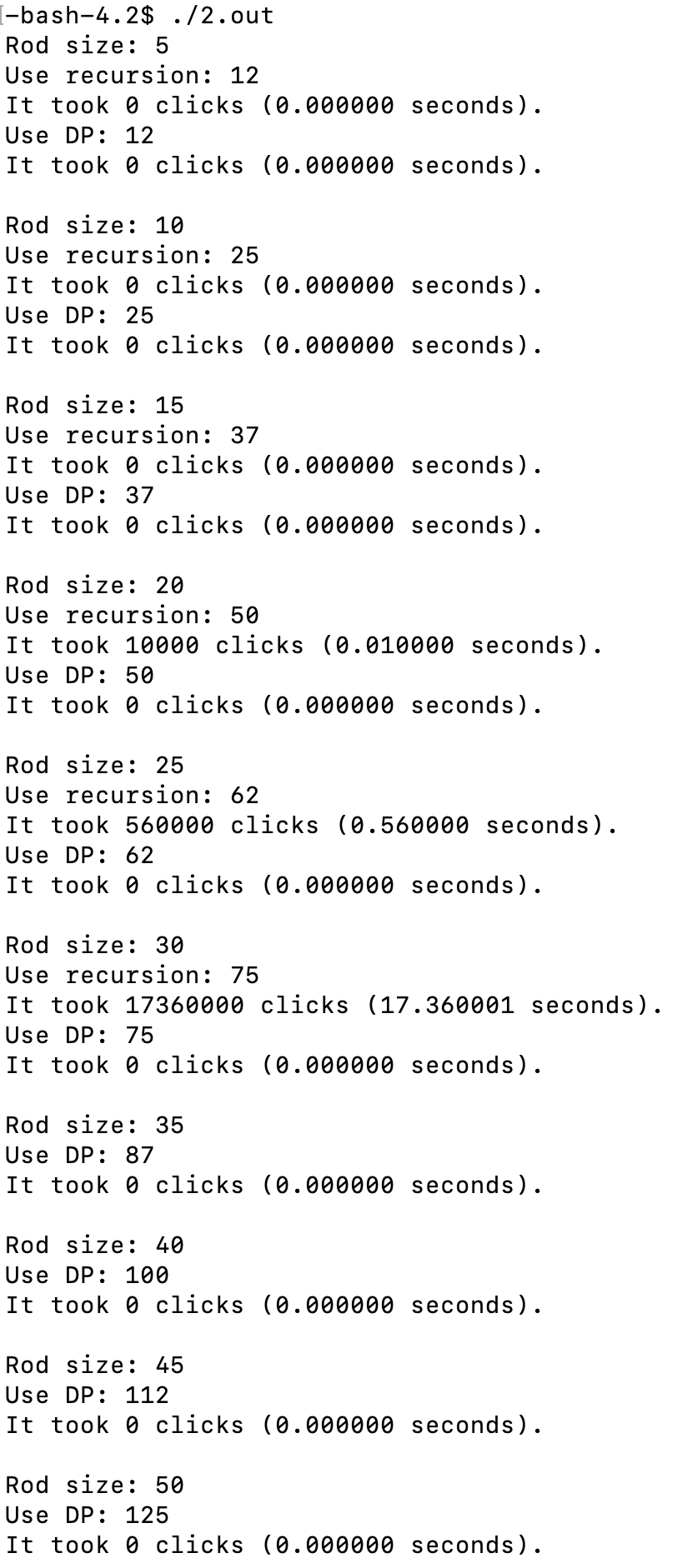
**Problem 1**

Result in COE Linux:



**Problem 2**

Result in COE Linux:

****

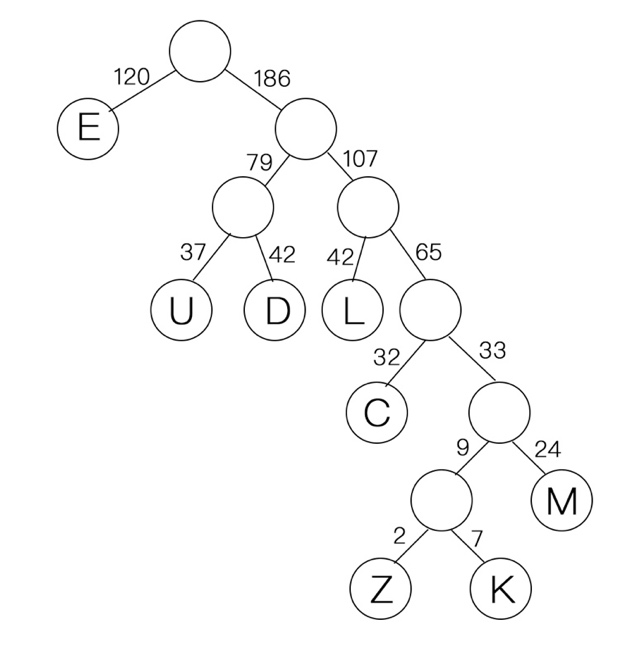
When rod size is 35, the running time is more than 500s, so recursive is not used in longer size.

Because COE Linux is so fast, the processing time is almost 0 when using DP in situation of 50. The following data is from Xcode IDE on my MacBook, which is relatively slower than COE Linux, but it still shows that DP is faster than Recursive.

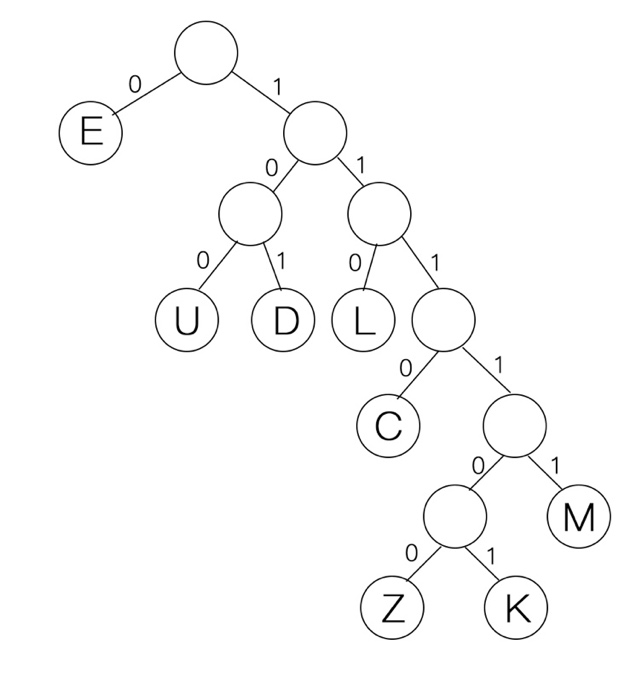
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rod Size** | **Recursive Time (s)** | **Recursive Max Revenue** | **Dynamic Time (s)** | **Dynamic Max Revenue** |
| 5 | 0.000003 | 12 | 0.000002 | 12 |
| 10 | 0.000023 | 25 | 0.000003 | 25 |
| 15 | 0.000719 | 37 | 0.000006 | 37 |
| 20 | 0.016127 | 50 | 0.000004 | 50 |
| 25 | 0.509955 | 62 | 0.000009 | 62 |
| 30 | 16.024006 | 75 | 0.000009 | 75 |
| 35 | >3 mins | / | 0.000009 | 87 |
| 40 | >3 mins | / | 0.000025 | 100 |
| 45 | >3 mins | / | 0.000021 | 112 |
| 50 | >3 mins | / | 0.000019 | 125 |

**Problem 3**

The following picture shows the weight display of the process of constructing a Huffman tree.



This picture shows the Huffman tree with the left branch of the weight changed to 0 and the right branch changed to 1.



For the eight letters to be coded with 0 or 1 of the path from the root to the leaf, the following table can be obtained.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Letter | E | U | D | L | C | M | Z | K |
| Huffman code | 0 | 100 | 101 | 110 | 1110 | 11111 | 111100 | 111101 |