Chapter 15 Solusion

frc6.com

i@frc6.com

https://github.com/frc123/CLRS-code-solution

11/3/2021

15.1

15.1-1

Proof. We prove by substitution method. For $n = 0$, $T(0) = 2^0 = 1$. For $n > 0$,

$$T(n) = 1 + \sum_{j=0}^{n-1} T(j) = 1 + \sum_{j=0}^{n-1} 2^j = 1 + (2^n - 1) = 2^n$$ □

15.1-2

Consider the following case:

length i	1	2	3
price p_i	1	6	8
density p_i/i	1	3	2.67

If we use "greedy" strategy, our solution will be "2 1", and the total price will be 7. However, the optimal way is "3", and the total price is 8.

15.1-3

```
1       /**
2       * running time: O(n^2)
3       * p: table of prices (index start from 0)
4       * n: length of rod
5       * c: cost of each cut
6       * return maximum revenue
7       */
8       int BottomUpCutRodWithCost(const std::vector<int>& p, int n, int c)
9       {
10          int *r, q, i, j;
11          r = new int[n + 1];
12          r[0] = 0;
```

```
13          for (j = 1; j <= n; ++j)
14          {
15              q = p[j - 1];
16              for (i = 0; i < j - 1; ++i)
17                  q = std::max(q, p[i] + r[j - i - 1] - c);
18              r[j] = q;
19          }
20          delete[] r;
21          return q;
22      }
```

15.1-4

```
1       /**
2       * p: table of prices (index start from 0)
3       * n: length of rod
4       * r: table of maximum revenue (index start from 1)
5       * s: table of optimal size i of the first piece to cut off (index start from 1)
6       * return maximum revenue
7       */
8       int ExtendedMemoizedCutRodAux(const std::vector<int>& p, int n, int *r, int *s)
9       {
10          int q, i, reminder_r;
11          if (r[n] >= 0) return r[n];
12          q = INT_MIN;
13          for (i = 0; i < n; ++i)
14          {
15              reminder_r = ExtendedMemoizedCutRodAux(p, n - i - 1, r, s);
16              if (q < p[i] + reminder_r)
17              {
18                  q = p[i] + reminder_r;
19                  s[n] = i + 1;
20              }
21          }
22          r[n] = q;
23          return q;
24      }
25
26      /**
27      * running time: O(n^2)
```

```
28      * p: table of prices (index start from 0)
29      * n: length of rod
30      * return (r, s)
31      * r: table of maximum revenue (index start from 1)
32      * s: table of optimal size i of the first piece to cut off (index start from 1)
33      * caller is responsible to deallocate return value r and s
34      */
35      std::pair<int*, int*> ExtendedMemoizedCutRod(const std::vector<int>& p, int n)
36      {
37          int *r, *s, i;
38          r = new int[n + 1];
39          s = new int[n + 1];
40          r[0] = 0;
41          s[0] = 0;
42          for (i = 1; i <= n; ++i) r[i] = INT_MIN;
43          ExtendedMemoizedCutRodAux(p, n, r, s);
44          return std::make_pair(r, s);
45      }
```

15.1-5

```
1       /**
2       * running time: O(n)
3       * n: n-th fibonacci number (must greater than 0)
4       */
5       int FibonacciNumber(int n)
6       {
7           int *f, i, result;
8           f = new int[n + 1];
9           f[0] = 0;
10          f[1] = 1;
11          for (i = 2; i <= n; ++i)
12              f[i] = f[i - 1] + f[i - 2];
13          result = f[n];
14          delete[] f;
15          return result;
16      }
```