Fibonacci number

Coin change problem

Given the denominations **c**: $c_1, c_2, \ldots, c_d$, the recurrence relation is:

$$\text{minNumCoins}(M) = \min_{\text{of}} \begin{cases} \text{minNumCoins}(M-c_1) + 1 \\ \text{minNumCoins}(M-c_2) + 1 \\ \ldots \\ \text{minNumCoins}(M-c_d) + 1 \end{cases}$$

```
RecursiveChange(M, c, d)
  if M = 0
    return 0
  bestNumCoins ← infinity
  for i ← 1 to d
    if M ≥ c_i
      numCoins ← RecursiveChange(M – c_i , c, d)
      if numCoins + 1 < bestNumCoins
        bestNumCoins ← numCoins + 1
  return bestNumCoins
```

```
DPChange(M, c, d)
  bestNumCoins_0 ← 0
  for m ← 1 to M
    bestNumCoins_m ← infinity
    for i ← 1 to d
      if m ≥ c_i
        if bestNumCoins_{m – c_i} + 1 < bestNumCoins_m
          bestNumCoins_m ← bestNumCoins_{m – c_i} + 1
  return bestNumCoins_M
```

Running time: $O(M*d)$

Rod cutting

The ***rod-cutting problem*** is the following. Given a rod of length n inches and a table of prices p_i for $i = 1, 2, \ldots, n$, determine the maximum revenue r_n obtainable by cutting up the rod and selling the pieces. Note that if the price p_n for a rod of length n is large enough, an optimal solution may require no cutting at all.

Consider the case when $n = 4$. Figure 15.2 shows all the ways to cut up a rod of 4 inches in length, including the way with no cuts at all. We see that cutting a 4-inch rod into two 2-inch pieces produces revenue $p_2 + p_2 = 5 + 5 = 10$, which is optimal.

We can cut up a rod of length n in 2^{n-1} different ways, since we have an independent option of cutting, or not cutting, at distance i inches from the left end,

length i	1	2	3	4	5	6	7	8	9	10
price p_i	1	5	8	9	10	17	17	20	24	30

```
CUT-ROD(p, n)
if n == 0
    return 0
q = −∞
for i = 1 to n
    q = max(q, p[i] + CUT-ROD(p, n − i))
return q
```

```
BOTTOM-UP-CUT-ROD(p, n)
let r[0..n] be a new array
r[0] = 0
for j = 1 to n
    q = −∞
    for i = 1 to j
        q = max(q, p[i] + r[j − i])
    r[j] = q
return r[n]
```

0-1 Knapsack

$$P[i,w] = \begin{cases} P[i-1,w] & \text{if } w < w_i \\ \max\{v_i + P[i-1, w - w_i], P[i-1,w]\} & \text{else} \end{cases}$$

	0	1			$w - w_i$			w			W
0	0	0	0	0	0	0	0	0	0	0	0
	0										
	0										
i-1	0										
i	0										
	0										
n	0										

Running time: $O(n*W)$

```
for w = 0 to W
  P[0, w] = 0
for i = 0 to n
  P[i, 0] = 0
  for w = 0 to W
      if w_i <= w // item i can be part of the solution
           if (v_i + P[i-1, w-w_i] > P[i-1, w])
                P[i, w] = v_i + P[i-1, w- w_i]
           else
                P[i, w] = P[i-1, w]
      else P[i, w] = P[i-1, w]  // w_i > w
```

Subset sum problem

You are given an array A and a number N. You need to find out if N is a sum of any subset of A or not.

Example:

A = {2, 4, 5, 6, 8}, N = 15 → True {4, 5, 6}

A = {2, 4, 5, 6, 8}, N = 0 → True {}

A = {2, 4, 5, 6, 8}, N = 3 → False

Hint: similar to 0-1 knapsack. Think of A as set of items, N as knapsack capacity.

Longest common subsequence

Given two strings x and y, find the longest common subsequence and its length.

Example:

x = “A**BCB**D**A**B”

y = “**B**D**C**A**BA**”

longest common subsequence = “BCBA”

longest common subsequence length = 4

x = “**AB**B**ACQ**”

y = “X**A**YZM**B**NN**A**LQ**C**TR**Q**”

longest common subsequence = “ABACQ”

longest common subsequence length = 5

Hint: CLRS 15.4

```
LCS-LENGTH(X, Y)
m = X.length
n = Y.length
let b[1..m, 1..n] and c[0..m, 0..n] be new tables
for i = 1 to m
    c[i, 0] = 0
for j = 0 to n
    c[0, j] = 0
for i = 1 to m
    for j = 1 to n
        if x_i == y_j
            c[i, j] = c[i − 1, j − 1] + 1
            b[i, j] = "↖"
        elseif c[i − 1, j] ≥ c[i, j − 1]
            c[i, j] = c[i − 1, j]
            b[i, j] = "↑"
        else c[i, j] = c[i, j − 1]
            b[i, j] = "←"
return c and b
```

```
PRINT-LCS(b, X, i, j)
if i == 0 or j == 0
    return
if b[i, j] == "↖"
    PRINT-LCS(b, X, i − 1, j − 1)
    print x_i
elseif b[i, j] == "↑"
    PRINT-LCS(b, X, i − 1, j)
else PRINT-LCS(b, X, i, j − 1)
```