Exercise 6.1 We distinguish the 6 steps:

(1) The hint suggests to use a state $S(j)$ which is the value of the maximum contiguous subsequence ending in the j-th number of the list.

(2) If $a_j \geq 0$ for at least one j then the optimal value is $\max_j S(j)$.
If $a_j < 0$ for all j then the optimal solution is the empty sequence, which has value zero by definition.
In short, the optimal value is $\max\{0, \max_j S(j)\}$.

(3) Initial value: $S(1) = a_1$.

(4) The recursion :

$$\begin{array}{lcll} S(j+1) & = & S(j) + a_{j+1} & \text{if } S(j) \geq 0, \text{ and} \\ S(j+1) & = & a_{j+1} & \text{if } S(j) \leq 0. \end{array}$$

Equivalently, $S(j+1) = \max\{a_{j+1}, S(j) + a_{j+1}\}$ for $j = 1, \ldots, n-1$.

(5) In the DP table we store $S_1, S_2, \ldots, S_n$ so the size of the table is $O(n)$. However, you can do with less space since you only need to know S_j and a_{j+1} to compute S_{j+1}. Also, we updte the maximum value S_j over the first j numbers. The total work space is needed is only $O(1)$.

(6) The running time is $O(n)$ since computing one subproblem takes $O(1)$ time and there are $O(n)$ subproblems.

Exercise 6.2

(1) Define $f(i)$ as the minimum cost for traveling from location 1 to location i.

(2) The optimal value is given by $f(n)$.

(3) $f(1) = 0$ (Cost for going from point 1 to point 1).

(4) The following recursion holds for $2 \leq i \leq n$:

$$f(i) = \min_j \{f(j) + (200 - (a_i - a_j))^2 \mid 1 \leq j < i\}.$$

(5) The size of the DP table is $O(n)$. Unlike the previous exercise, we need to keep all values in memory since to compute $f(i)$ we need to know all $f(j)$ for $j < i$. The work space needed is $O(n)$.

(6) The time used is $O(n^2)$ since it takes $O(n)$ time to compute each if the $O(n)$ values.

Exercise 6.3

(1) Let $f(i)$ be the maximum profit over the first i locations when we open location i ($1 \leq i \leq n$).

(2) The optimal value is $\max\{f(i) \mid 1 \leq i \leq n\}$.

(3) $f(1) = p_1$.

(4) $f(i) = p_i + \max_j\{f(j) \mid m_i - m_j \geq k\}$.

(5) The size of the DP-table is $O(n)$.

(6) The time to compute one value in the table is $O(n)$. Hence, the total time is $O(n^2)$.

A slightly different DP is as follows. We remove the restriction in the subproblem that location i must be opened.

(1) Let $f(i)$ be the maximum profit over the first i locations ($1 \leq i \leq n$).

(2) The optimal value is $f(n)$.

(3) $f(1) = p_1$.

(4) $f(i) = \max\{f(i-1),\ p_i + \max_j\{f(j) \mid m_i - m_j \geq k\}\}$.

(5) The size of the DP-table is $O(n)$.

(6) The time to compute one value in the table is $O(n)$ so the total time is $O(n^2)$.

Exercise 6.4 Say that a substring is *valid* if it is a sequence of words.

(1) Define $f(i) =$ true if $s[1..i]$ is a valid substring and let $f(i) =$ false otherwise.

(2) s is a valid string iff $f(n) = \text{true}$.

(3) $f(0) = \text{true}$.

(4) For $i = 1 \ldots n$, let $f(i) = \text{true}$ if there is a $j \in \{0, \ldots, i-1\}$ such that $f(j) = \text{true}$ and $s[j+1 \ldots i]$ is a valid word.
$f(i) = \text{false}$ otherwise.

(5) Size of DP table is $O(n)$.

(6) The time to compute one value in the table is $O(n)$. Hence, the total time is $O(n^2)$.

Exercise 6.5 (a) There are 8 different patterns for a single column. (A '1' means a pebble is placed at that position.)

0	1	0	0	0	1	0	1
0	0	1	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	0	0	1	0	1	1

(b)
Denote the 8 different patterns by *type* $t = 1, 2, \ldots, 8$.

(1) Let $f(k, t)$ be the total profit that can be obtained from the first k columns under the restriction that the k-th column is of type t ($k = 1, \ldots, n$ and $t = 1, \ldots, 8$).

(2) The optimal value is $\max\{f(n, t) \mid 1 \leq t \leq 8\}$.

(3) Let $\text{Profit}(k, t)$ be the profit obtained from column k if pattern t is chosen. Then, the initial values are $f(1, t) = \text{Profit}(1, t)$ for all t.

(4) For each type t and $2 \leq k \leq n$ the recursion is

$$f(k, t) = \text{Profit}(k, t) + \max_s \{f(k-1, s) \mid s \text{ compatible with } t\}$$

(5) The number of subproblems is $O(n)$ but the workspace needed to compute the optimal value is only $O(1)$. However, if we want to find not just the optimal value but the optimal solution then we need $O(n)$ for the space since the optimal solution is of size $O(n)$.

(6) The time to compute one value in the DP-table is $O(1)$. So the total time is $O(n)$.

NB. In general, when we have m rows, the number of types is $2^{\Omega(m)}$. (That means, at least 2^{cm} for some constant $c > 0$). Then, the running time of this DP-approach is not polynomial any more.

Exercise 6.6 Let $s = s_1 s_2 \dots s_n$ be the string and denote by $s[i..j]$ the substring $s_i s_{i+1} \dots s_j$.

(1) Define $f(i, j, v)$ = true iff substring $s[i..j]$ can be turned into v, where $v \in \{a, b, c\}$ and $1 \leq i \leq j \leq n$.

(2) The answer is 'yes' iff $f(1, n, a) =$ true.

(3) $f(i, i, v) =$ true iff $s_i = v$, where $v \in \{a, b, c\}$ and $1 \leq i \leq n$.

(4) $f(i, j, v) =$ true iff there is some $k \in \{i, \dots, j-1\}$ and $v_1, v_2 \in \{a, b, c\}$ such that:

- $f(i, k, v_1) =$ true and
- $f(k+1, j, v_2) =$ true and
- $v_1 v_2 = v$.

(5) The size of the table is $O(n^2)$.

(6) The time to compute one value in the DP-table is $O(n)$. So the total time is $O(n^3)$.

Exercise 6.7 Let $x = x_1 x_2 \dots x_n$ be the string and denote by $x[i..j]$ the substring $x_i x_{i+1} \dots x_j$. Define $x[i..j]$ as the empty string if $j < i$.

(1) Define $f(i, j)$ as the length of the longest palindrome in substring $x[i..j]$.

(2) The optimal value is $f(1, n)$.

(3) $f(i, i) = 1$ and $f(i, i-1) = 0$ for all $i \in \{1, \dots, n\}$.
(We need these initial values for step 4)

(4) The recursion for $1 \leq i < j \leq n$ is as follows.

if $x_i = x_j$:
$$f(i, j) = 2 + f(i+1, j-1)$$
else:
$$f(i, j) = \max\{f(i, j-1), f(i+1, j)\}.$$

(5) The size of the table is $O(n^2)$.

(6) The time to compute one value in the DP-table is $O(1)$. So the total time is $O(n^2)$.

Exercise 6.8 Denote by $x[1..i]$ the substring $x_1x_2\dots x_i$ and denote by $y[1..j]$ the substring $y_1y_2\dots y_j$.

(1) Let $f(i,j)$ be the length of the longest common *suffix* of $x[1..i]$ and $y[1..j]$.

(2) The optimal value is $\max_{i,j}\{f(i,j) \mid 1 \le i \le n, 1 \le j \le m\}$.

(3) $f(i,j) = 0$ if $i = 0$ or $j = 0$.

(4) The recursion for $i \in \{1,\dots,n\}$ and $j \in \{1,\dots,m\}$ is

if $x_i = y_j$:
$$f(i,j) = 1 + f(i-1, j-1)$$
else:
$$f(i,j) = 0.$$

(5) The size of the table is $O(n^2)$.

(6) The time to compute one value in the DP-table is $O(1)$. So the total time is $O(n^2)$.

Exercise 6.9

(1) Let $f(i,j)$ be the minimum cost for cutting a string of length i exactly j times ($j < i$).

(2) The optimal value is $f(n,m)$.

(3) $f(1,0) = 0$.

(4) To compute $f(i,j)$ we try all possible values k such that the first cut splits the string in strings of length k and $i-k$. Further, we try all possible values h for the number of cuts in the string of length k. Then, the other string is cut $j-h-1$ times.

$$f(i,j) = i + \min_{k,h}\{f(k,h) + f(i-k, j-h-1)\},$$

where the minimum is taken over all k and h such that:

- $1 \leq k \leq i-1$,
  (cut in two strings of length k and $i-k$.)
- $0 \leq h \leq k-1\}$,
  (a string of length k can be cut at most $k-1$ times.)
- $0 \leq j-h-1 \leq i-k-1\}$
  (a string of length $i-k$ can be cut at most $i-k-1$ times.)

(5) The size of the table is $O(nm)$.

(6) The time to compute one value in the DP-table is $O(nm)$. So the total time is $O(n^2m^2)$.

Exercise 6.10

(1) $f(i,j)$ is the probability of exactly j times head among the first i coins $(0 \leq j \leq i \leq n)$.

(2) The value we want to compute is $f(n,k)$.

(3) $f(1,1) = p_1$ and $f(1,0) = 1-p_1$.

(4)
$$f(i,j) = p_i \cdot f(i-1,j-1) + (1-p_i) \cdot f(i-1,j).$$
Explanation: If coin i flips head then the other $i-1$ coins must flip head exactly $j-1$ times. If coin i flips tails then the other $i-1$ coins must flip head exactly j times.

(5) The size of the table is $O(nk)$.

(6) The time to compute one value is $O(1)$ so the total time is $O(nk) = O(n^2)$.