Read Chapter 6 of Dasgupta. The chapter is basically a collection of examples of Dynamic Programming: some are easy, some are quite difficult. The best way to learn DP is to read many examples and to make DP's yourself. Chapter 6 contains many exercises: too many to discuss all of them in this course. As you may notice, most of these exercises do not refer directly to the examples given in the chapter. So the message is: You do not need to memorize or even understand all the examples given but it is certainly good to read them.

The idea of DP is always the same: A problems is solved by solving (smaller) subproblems. Solutions to subproblems are stored in memory (the DP table). To solve a subproblem we make use of the stored information on other subproblems. In building and analyzing a DP ask yourself the following questions:

(1) What is the subproblem to solve? In other words: What will be in the table?

(2) What is the optimal value, expressed in terms of the subproblems? In other words: How do you find the optimal value once the table is filled?

(3) What are the initial values? In other words: What values can you fill in right away?

(4) What is the recurrence used? In other words: Given the initial values, how to compute the rest?

(5) What is the used space? This is often the size of the DP table. For example, $O(n)$ or $O(n^2)$. But sometimes we can do with less space, see for example Exercise 1.

(6) What is the running time? This is usually (but not always) the size of the table times the time it takes to compute one value of the table.

Next to a value for each subproblem one usually stores in the DP-table a corresponding solution or just some pointer from which a solution can be restored. For example, in Dijkstra's algorithm to compute a shortest path from s to t there is a subproblem for each $v \in V$ which is to find a shortest paths from s to v. When the length of a shortest path from s to v has been computed, we can either store this path or just a pointer to the vertex u that comes before v on this path since that is enough to restore the shortest path once the DP-table has been filled.

NB. In the exercises we shall only care about computing the value of the optimal solution and not about the solution itself since that one can always be found easily with some extra bookkeeping.

Exercises

All exercises from Chapter 6 are nice examples of DP but we shall restrict to the first ten: **6.1–6.10**.
To get started, the answer to 6.1 is provided.

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}. So 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.