CS161

1 Divide-and-conquer

We can solve a problem recursively as follows:

- Divide the problem into a number of subproblems.
- Conquer the subproblems by solving them recursively.
- Combine the solutions to the subproblems.

A classic example is the maximum-subarray problem. Given a list of prices over an N-day period, you have the option to buy once and sell once. What is the optimal strategy?

Brute force: just search over the $n2$ possible pairs.

Alternatively, just break down the maximum sub-array problem into three cases:

- Left subarray case,
- Right subarray case,
- Cross subarray case.

2 Dynamic programming

Recall that divide-and-conquer algorithms partition the problem into disjoint subproblems, solve the subproblems recursively, and then combine their solutions to solve the original problem.

In constant, dynamic programming applies when the subproblems overlap. A DP algorithm solves each subsubproblem juts once and saves it answer in a table. Most often, DP applies to optimization problems.

Example. Given a rod of length n inches and a tbable of prices p_i for $i = 1, \ldots, n$, determine the maximize revenue r_n obtained by cutting the rod and selling the pieces.

The idea is to use the fact that $r_n = \max_{1 \leq i \leq n}(p_i + r_{n-i})$. We can calculate this either bottom-up or top-down. Bottom-up algorithms sometimes have better constant factors.

3 Max flow algorithms