Running Lecture Outline: Discrete Optimization go brr

[Chirayu Salgarkar]

Summer 2024

Contents

1 Week 2

1.1 Knapsack Problem

The knapsack problem is a conventional dynamic programming problem.

Let's start with the 01 knapsack problem. We have a set of objects, with a defined weight. Let's give some numbers here.

We begin with $n = 4$ objects, each with some profit P and weight ω. We also have a bag of capacity $m = 8$.

$$P = \{1, 2, 5, 6\}$$

$$\omega = \{2, 3, 4, 5\}$$

Objective: fill bag with above objects.

What's a good output for this? Let's define x_i such that if it's included, 1 is output, but if it's not included, 0 is output.

For instance, a valid output here would be something like:

$$\{1, 0, 0, 0\}$$

Therefore, this is basically an optimization problem. Well, clearly, it is.

Here's the optimization problem in nicer detail:

$$\max \sum p_i x_i$$

such that

$$\sum \omega_i x_i \leq m$$

Let's talk about dynamic programming now. Dynamic programming works under a paradigm of *seek and solve* solutions. Dynamic programming then asks you to evaluate all solutions, and then pick to most optimal (best) one for the task at hand.

What's the number of solutions to this problem? Clearly, an upper bound is 2^n. $\mathcal{O}(2^n)$ is not good. Dynamic programming does $\mathcal{O}(2^n)$ indirectly.

Abdul Bari uses a tabular method here. I highly recommend his video here: the video is called *4.5 0/1 Knapsack - Two Methods - Dynamic Programming*. Here's the TiX filled out.

		0	1	2	3	4	5	6	7	8
P_i	ω_i	0	0	0	0	0	0	0	0	0
1	2	0	0	1	1	1	1	1	1	1
2	3	0	0	1	2	2	3	3	3	3
5	4	0	0	1	2	5	6	6	7	8
6	5	0	0	1	2	5	6	6	7	8

Take a minute and read it, see if you understand what's going on here.

There's a formula here as well, that's pretty slick:

$$V[i, \omega] = \max\{V[i-1, \omega], V[i-1, \omega - \omega[i]] + P[i]\}$$

Verify for yourself.

Now, we need to find x_1, x_2, x_3, x_4. This is the *sequence of decisions* part.