Dynamic Programming! (CLRS Chapter 15, KT Chapter 6)

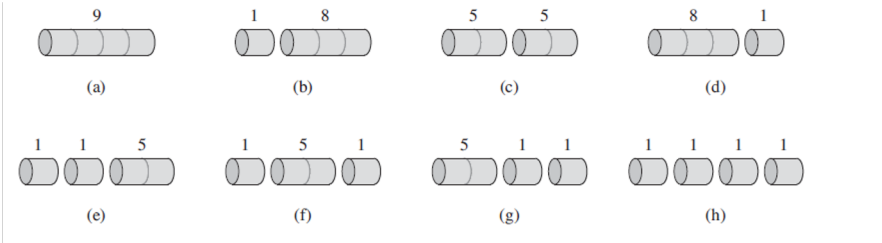
1 Intro (Recap)

- *Dynamic programming* is a technique for designing algorithms.
  - Fits problems where we want to find an *optimal solution.*
  - Unlike sorting or searching.
- Based on *solving smaller instances of the same problem.*
  - Similar to *divide and conquer* except that the smaller problems overlap.

2 Rod cutting (CLRS §15.1)

Recall: What is the highest revenue we can get by cutting an n foot rod and selling the pieces?

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



Example *Maximize revenue from rod of length 5.*

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

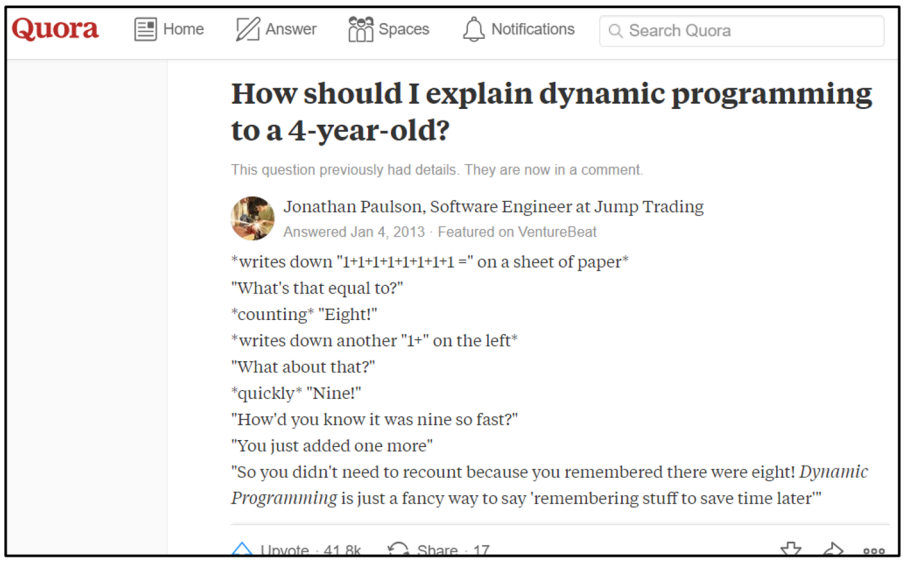
Last time we discussed an algorithm for finding optimal revenue. How do we output the actual sequence of cuts we should do?

```
1  r[0] = 0
2  for j = 1 to n
3       q =
4       for i = 1 to
5            if q
6                 q =
7                 s[j] =
8       r[j] =
9  print r[n]
10 while n > 0
11      print
12      n =
```

3 Dynamic Programming Principles (CLRS §15.3)

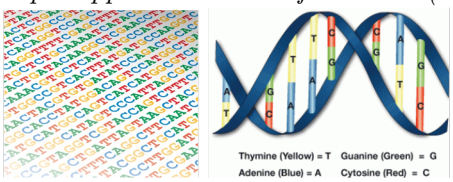
The Dynamic Programming Approach

- Instead of one problem,
- First of the problem.
- Gradually handle by relying on smaller ones.
- in the calculations.



4 Longest common subsequence (CLRS §15.4)

Example application: bioinformatics (similarity between DNA sequences)



Input:

Output:

Example

- Subproblems:

- Guess:

- Recurrence:

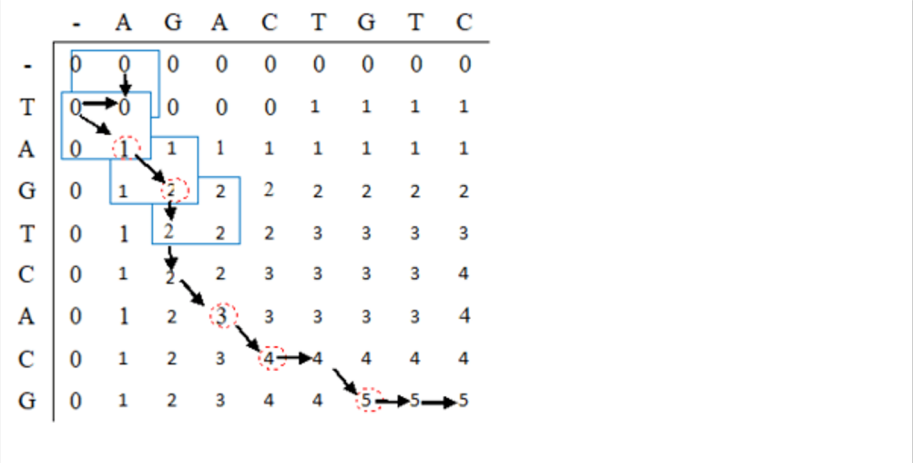
4.1 Algorithm idea

Example

Bottom up algorithm:

```
for i = 1 to m:
    C[i, 0] =
for j = 0 to n:
    C[0, j] =
for i = 1 to m:
    for j = 1 to n:
        if
            C[i, j] =
            B[i, j] =
        else if
            C[i, j] =
            B[i, j] =
        else
            C[i, j] =
            B[i, j] =
return
```

Example


- A G A C T G T C
- 0 0 0 0 0 0 0 0 0
T 0 0 0 0 0 1 1 1 1
A 0 1 1 1 1 1 1 1 1
G 0 1 2 2 2 2 2 2 2
T 0 1 2 2 2 3 3 3 3
C 0 1 2 2 3 3 3 3 4
A 0 1 2 3 3 3 3 3 4
C 0 1 2 3 4 4 4 4 4
G 0 1 2 3 4 4 5 5 5

General Guidelines
How do we determine if a problem can be handled using dynamic programming?

- The problem asks to find an optimal solution
- It is easy to obtain an optimal solution given solutions to slightly smaller subproblems.
- Likely to lead to an efficient algorithm when the subproblems overlap.

"I spent the Fall quarter of 1950 at RAND… We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word research… he would turn red, and he would get violent if people used the term research in his presence. You can imagine how he felt, then, about the term mathematical…

I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. In the first place I was interested in planning, in decision making, in thinking… I decided therefore to use the word programming. I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying… Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities." --- Richard Bellman.

5 steps for DP

1.

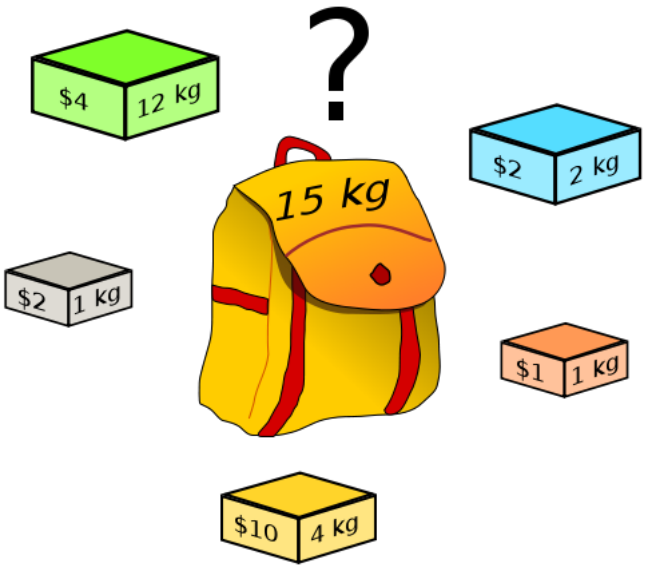
2.

3.

4.

5.

5 Knapsack (KT §6.4)



- List of n items each of
- Knapsack of size
- What is the max total

1. Subproblems

2. Guess

3. Recurrence

Example