CS3000
5/27 Tues.

Admin

- exam #1 graded!
  - ↳ regrade reqs by 6/1 9pm
- HW3 due Fri 9pm
- Rec 3 today

Agenda

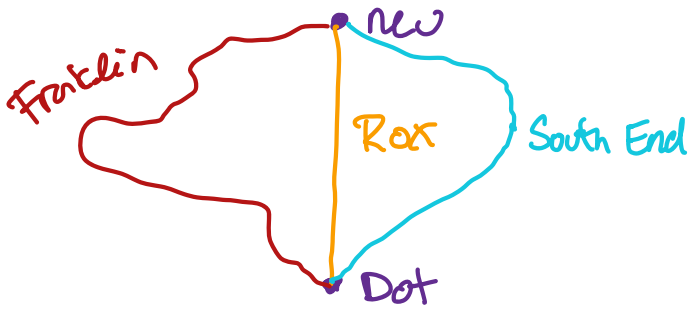
1. Dynamic Programming
2. Recursion ⇒ DP
3. DP Solution

\includegraphics{ ~ .png}

1. Dynamic Programming

↳ How do we solve an optimization problem?

(ex) Larry biking home from work



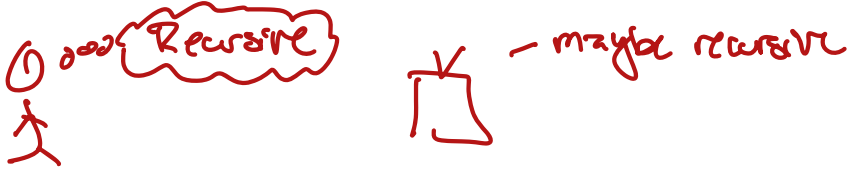
- 3 possible routes
- all go from new to Dot
- "valid" solutions

What's the best route?

- least time
- fewest hills
- most safe
- most enjoyable
- fewest intersections
- easiest side streets

Find the optimal solution

- **Brute force**: compute all solutions, keep the best one
  - ↳ can be intractable!
    - exponential or worse
- **Recursion**: solve smaller subproblems
  - ↳ can we do better?
    - 1. D+C
    - 2. Data Structure
    - 3. Dynamic Programming ☆



DP
- optimization problems
- solve subproblems
- combine smaller solutions
- re-use smaller solutions

subproblems overlap
↳ o/w, no point in DP!

Approach:
1. Define a recursive equation
2. when computing solution to subproblem...
   - if we've already solved, re-use it!
   - otherwise, save it.

Solution
1. value of optimal solution
2. actual optimal solution

2. Recursion ⇒ DP

↳ ex problem: Rod Cutting :)

- cut into pieces
- sell the pieces

- Rod of length n
- Array p of prices
- cut rod into
  $1 \le k \le n$ pieces

$p = \langle 1, 5, 8, 7 \rangle$
(1 2 3 4)

$n = 4$

length 1 $1
length 2 $5
length 3 $8
length 4 $7

- $n = i_1 + i_2 + \dots + i_k$
- p_{ik} = price for piece i

i_1 = length of piece 1
⋮
i_k = length of piece k

- Goal: maximize revenue

$r_n = p_{i_1} + p_{i_2} + \dots + p_{ik}$

Solution
- value of optimal solution: r_n
- actual optimal solution: what cuts do we make?

ex: valid solution
↳ cut into $\langle 1, 1, 2 \rangle$

$n = 4$
$k = 3$

(1 2 3 4)
$p = \langle 1, 5, 8, 7 \rangle$

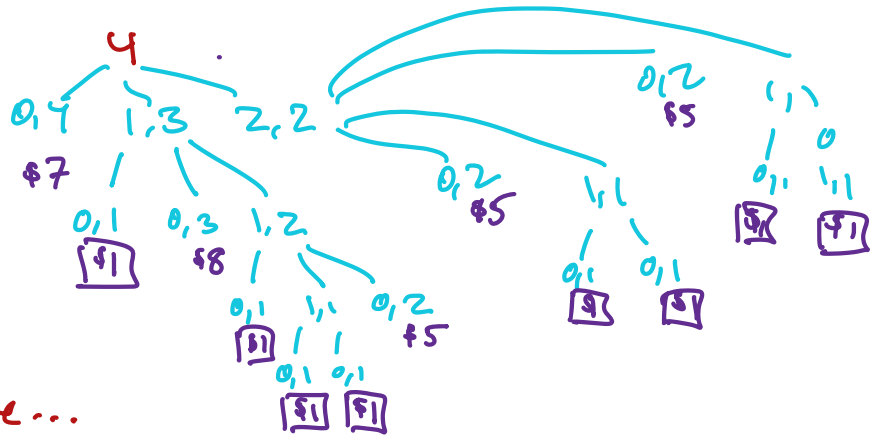
$r_n = p_{i_1} + p_{i_2} + p_{i_3}$
$= \$1 + \$1 + \$5$
$= \boxed{\$7}$

but is it optimal?

value of this solution is $7

$p = \langle 1, 5, 8, 7 \rangle$
1 2 3 4

- break into smaller subproblems



- How many times do we compute...

0, 4	1
0, 3	1
0, 2	3
0, 1	8

- Recursion gives an optimal solution
- But, do extra work!
  - ↳ Recursive, Brute Force: $\Theta(2^n)$
- ↳ $r_n = \$10$ (value of optimal soln)

Recursive Approach

- Base case = 0, \$0
- Optimal for length 3?
  - ↳ max of \$8, or optimal for $\langle 1, 2 \rangle$
    - ↳ optimize for 2? max of \$5 or $\langle 1, 1 \rangle$

High level:

CUT(p, n)
- base case check
- for i = 1 to n
  - compute max p_n, $p[i] + CUT(p, n-i)$

Recursive Equation

$$r_n = \max\{p_i + r_{n-i} : 1 \le i \le n\}$$

base case: $r_0 = 0$

→ break into subproblems
length n
... keep at n?
... solve smaller versions?

12:39

3. DP Solution

Approach...
1. Recursive equation
2. when solving subproblems...
   - if already solved, use it!
   - else, solve and save!

Rod-cutting recursion...
- rod of length n...
  - keep it length n?
  - or, break into
    $1, n-1$
    $2, n-2$
    ...
    → smaller subproblems

→ Equation

$r_n = \max\{p_i + r_{n-i} : 1 \le i \le n\}$

base case $r_0 = 0$

Memoization
↳ save the answers, so you don't need to recompute them

Top Down
↳ still has recursive calls
don't make it if we don't need to!

store solutions to subproblems in array r $[-\infty, -\infty, -\infty \ldots, -\infty]$

```
MEMOIZED-CUT-ROD(p, n)
1  let r[0 : n] be a new array
2  for i = 0 to n
3      r[i] = −∞
4  return MEMOIZED-CUT-ROD-AUX(p, n, r)
```

- $r[0]$ base case, $r[1:n]$ optimal solution
  $[-\infty, -\infty, \ldots -\infty]$

p $\langle 1, 5, 8, 7 \rangle$ $n = 4$
1 2 3 4

① $n=4$
base case? no
↳ keep at length 4?
or, break into $1, 3$
$2, 2$
→ ② $n=3$
keep length 3?
or, break into $1, 2$ → ③ $n=2$
keep length 2?
or, break into $1, 1$

$r = \langle -\infty, -\infty, -\infty, -8, -\infty \rangle$

↳ $r[0] = 0$

if $r[n]$ has been computed, return it
if r_n is computed for first time, save in $r[n]$
(also return it)

$r_n = \max\{p_i + r_{n-i} : 1 \le i \le n\}$

```
MEMOIZED-CUT-ROD-AUX(p, n, r)
1  if r[n] ≥ 0
2      return r[n]
3  if n == 0
4      q = 0
5  else
6      q = −∞
7      for i = 1 to n
8          q = max{q, p[i] + MEMOIZED-CUT-ROD-AUX(p, n − i, r)}
9  r[n] = q
10 return q
```

> Have we solved it?

> Base case, put q in r[0]

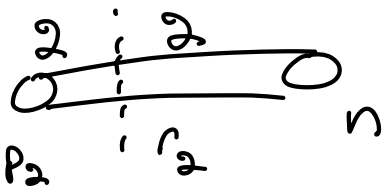
→ make recursive calls
- p, r are same
- n decreases

Keep rod, or break into

p_i vs. r_{n-i}

↳ price for length i

↳ recursive r_{n-i}



r by the end... $\langle 0, \$1, \$5, \$8, \$10 \rangle$
0 1 2 3 4

↳ value of optimal soln for every subproblem! :)