Lec 22 Dynamic Programming (DP)

Idea: solving subinstances and saving results in useful way

General structure

0. Start with some decision / optimisation / counting problem

   things like data transformation may not be one of these

   decision: is there path with len at most..

   optimisation: find shortest path

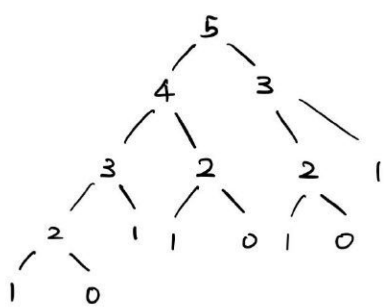
   counting: how many paths with given property

1. Develop recursive solution
2. Recognise how to reuse results from subinstances
3. Count num of unique subinstances for analysis
4. Implement
   - Memoisation
   - Bottom-up

Fibonacci example

fib(n) = if (n ≤ 1) then 1 else fib(n-2) + fib(n-1)

Call tree fibb 5

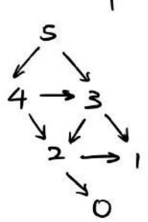


work $O(\phi^n)$ $\phi = \frac{\sqrt{5}+1}{2}$

span $O(n)$

Bad :C

Result dependency



So we need 6 = n+1 unique instances

work $O(n)$

span $O(n)$

Bottom-up Impl

```
fib n =
  let
    loop a b k =
      if k = n then a
      else loop b (a+b) (k+1)
  in
    loop 1 1 0
  end
```

Subset sum problem (SS)   NP - hard

Given set $S \subseteq \mathbb{Z}^+$ and $k \in \mathbb{Z}^+$, is there $X \subseteq S$, $\sum_{x \in X} x = k$?

→ Actually the base for some crypto system that was broken

Even though NP-hard it's easy to find sol for some input.
Pseudopoly — polynomial to k, so if k itself poly to |S| we get poly to |S|

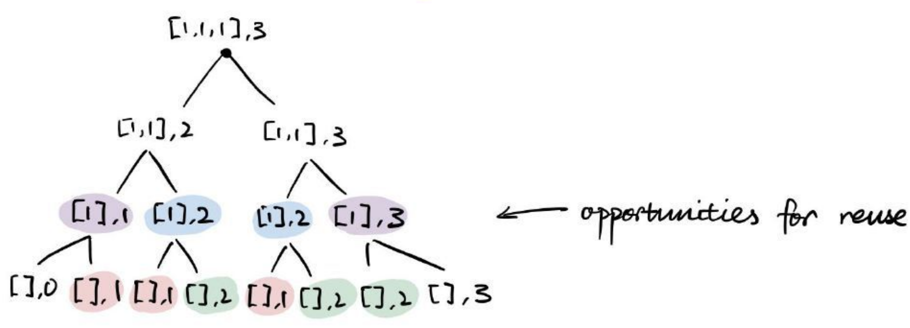
Recursive sol

hard to memoise as key

```
SS (S, k) = case (S,k) of
   (_, 0) => true
  | ([], _) => false
  | (x::xs, k) => if k < x then SS(xs, k) else
                  SS(xs, k-x) orelse SS(xs, k)
```

$W(n) = 2W(n-1) + O(1)$ exponential !

Ex. $S = [1,1,1]$ $k = 3$



$k' \in \{0, \dots, k\}$, $|S'| \in \{0, \dots, |S|\}$
so num unique subinstances is $(|S|+1)(k+1)$

If reuse results, work $O(|S|k)$
span $O(|S|+1)$

Representing lookup table

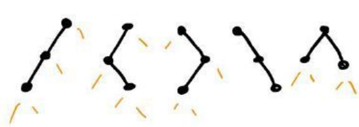
Our table wants to have subinstances as key, but if input has list how to hash / compare list? Expensive!

In practice try convert subinstance to integer

```
SS (S, k) =
  let
    n = |S|                          use this as key to table, or even 2D array
    SS' (i, k') = (case (i, k') of
      (_, 0) => true
     | (n, _) => false                         should look up here
     | (i, k') => if (k < S[i]) then SS' (i+1, k) else
                  SS'(i+1, k'-S[i]) orelse SS' (i+1, k')
  in
    SS' (0, k)
  end
```

Counting problem example

Count number of rooted binary tree shape with size n

n	shapes	count
0	$\emptyset$	1
1		1
2		1
3		5

$$T(n) = \begin{cases} 1 & \text{if } n \le 1 \\ \sum\limits_{\substack{i=0 \\ \text{left subtree size}}}^{n-1} \underbrace{T(i)}_{\text{left subtree}} \underbrace{T(n-i-1)}_{\text{right subtree}} & \text{else} \end{cases}$$

Num unique subinstances = $n + 1$

Work per subinstance — $O(n)$ to do the sum

span — $O(\lg n)$ reduce op+

Overall work $\sum_{i=0}^{n} O(i) \in O(n^2)$

$\sum_{i=0}^{n} O(\lg i) \in O(n \lg n)$