Dynamic Programming 101

- DP = recursion (divide-n-conquer) + caching (overlapping subproblems)

- the simplest example is Fibonacci

\[
f(n) = f(n - 1) + f(n - 2) \\
f(1) = f(2) = 1
\]

- naive recursion without memoization: \(O(1.618...n)\)

```python
def fib(n):
    if n <= 2:
        return 1
    return fib(n-1) + fib(n-2)
```

- DP1: top-down with memoization: \(O(n)\)

```python
fibs={1:1, 2:1}
def fib1(n):
    if n not in fibs:
        a, b = fib1(n-1) + fib1(n-2)
        fibs[n] = a+b, a
    return fibs[n]
```

- DP2: bottom-up: \(O(n)\)

```python
def fib0(n):
    a, b = 1, 1
    for i in range(3, n+1):
        a, b = a+b, a
    return a
```
Number of Bitstrings

• number of \( n \)-bit strings that do not have 00 as a substring
  
  • e.g. \( n=1 \): 0, 1; \( n=2 \): 01, 10, 11; \( n=3 \): 010, 011, 101, 110, 111
  
  • what about \( n=0 \)?
  
  • first bit “1” followed by \( f(n-1) \) substrings
  
  • first two bits “01” followed by \( f(n-2) \) substrings

\[
f(n) = f(n - 1) + f(n - 2)
\]

\[f(1)=2, f(0)=1\]
Max Independent Set

- max weighted independent set on a linear-chain graph
- e.g. 7 -- 2 -- 3 -- 5 -- 8
- subproblem: $f(n) --\text{max independent set for } a[1]..a[n]$ (1-based index)
  \[ f(n) = \max\{f(n-1), f(n-2) + a[n]\} \]
  \[ f(0)=0; f(1)=a[1]\] better: $f(0)=0; f(-1)=0$

```python
def max_wis2(a):
    best, back = {-1: 0, -2: 0}, {} # 0-based index!
    n = len(a)
    for i in range(n):
        best[i] = max(best[i-1], best[i-2]+a[i])
        back[i] = best[i] == best[i-1]
    return best[n-1], solution(n-1, a, back)

def solution(i, a, back):
    if i < 0:
        return []
    return solution(i-1, a, back) if back[i] else (solution(i-2, a, back) + [a[i]])
```

```
Summary

- Dynamic Programming = divide-n-conquer + overlapping subproblems
- “distributivity” of work: \((a \otimes c) \oplus (b \otimes c) \oplus (a \otimes d) \oplus (b \otimes d) = (a \oplus b) \otimes (c \oplus d)\)
- two implementation styles
  - 1. recursive top-down + memoization
  - 2. bottom-up
- also need backtracking to recover best solution (recommended: backpointers)
- three steps in solving a DP problem
  - define the subproblem
  - recursive formula
  - base cases
Deeper Understanding of DP

- **divide-n-conquer**
  - single divide, independent conquer, combine

- **DP = divide-n-conquer with multiple divides**
  - for all possible divide
    - divide
    - conquer with memoization
    - combine subsolutions using the combination operator $\otimes$
  - summarize over all possible divides using summary operator $\oplus$

- multiple divides $\Rightarrow$ overlapping subproblems
  - each single divide $\Rightarrow$ independent subproblems!

<table>
<thead>
<tr>
<th></th>
<th>$\oplus$</th>
<th>$\otimes$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fib</strong></td>
<td>+</td>
<td>$\times$</td>
</tr>
<tr>
<td><strong>MIS</strong></td>
<td>max</td>
<td>+</td>
</tr>
<tr>
<td><strong># BSTs</strong></td>
<td>+</td>
<td>$\times$</td>
</tr>
<tr>
<td><strong>knapsack</strong></td>
<td>max</td>
<td>+</td>
</tr>
<tr>
<td><strong>shortest path</strong></td>
<td>min</td>
<td>+</td>
</tr>
</tbody>
</table>
## One-way vs. Two-way Divides

<table>
<thead>
<tr>
<th>Divide-n-conquer</th>
<th>Two-way (binary divide)</th>
<th>One-way (unary divide)</th>
</tr>
</thead>
<tbody>
<tr>
<td>quicksort, best-case</td>
<td>quicksort, worst-case</td>
<td></td>
</tr>
<tr>
<td>mergesort</td>
<td>quickselect</td>
<td></td>
</tr>
<tr>
<td>tree traversal (DFS)</td>
<td>binary search</td>
<td></td>
</tr>
<tr>
<td>heapify (top-down)</td>
<td>search in BST</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DP</th>
<th># of BSTs (hw5)</th>
<th>Fib, # of bitstrings (hw5)…</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>optimal BST</td>
<td>max indep. set (hw5)</td>
</tr>
<tr>
<td></td>
<td>RNA folding (hw10)</td>
<td>knapsack (all kinds, hw6)</td>
</tr>
<tr>
<td></td>
<td>context-free parsing</td>
<td>Viterbi (hw8)</td>
</tr>
<tr>
<td></td>
<td>matrix-chain multiplication, …</td>
<td>LCS, LIS, edit-distance,…</td>
</tr>
<tr>
<td></td>
<td>Two Divides</td>
<td>Multiple Divides</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td><strong>DP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>two divides</strong></td>
<td><strong>multiple divides</strong></td>
</tr>
<tr>
<td></td>
<td>Fib, # of bitstrings (hw5)…</td>
<td># of BSTs (hw5)</td>
</tr>
<tr>
<td></td>
<td>max indep. set (hw5)</td>
<td>unbounded knapsack (hw6)</td>
</tr>
<tr>
<td></td>
<td>0-1 knapsack (hw6)</td>
<td>bounded knapsack (hw6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Viterbi (hw8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RNA folding (hw10)</td>
</tr>
</tbody>
</table>