Chapter 5, Dynamic Programming:

- Introduction and sample problems (last week Friday)
- Principles of DP (Monday)
- DP algorithms, solutions to sample problems (**Today**)
- Optimal BSTs (Friday)
- *Review for Test 2 (next week Monday)*
- *No class (next week Wednesday)*
- *Test 2 (next week Thursday, in lab)*

Today:

- Recursive characterization and algorithm for Knapsack
- Recursive characterization and algorithm for Longest Common Subsequence
- Recursive characterization and algorithm for Matrix Multiplication

Ex 6.5. Explain why this function can't use memoization:

```
idgen = -1

def make_unique_id(name) :
    # global allows us to modify idgen inside this function
    global idgen
    idgen += 1
    return name + str(idgen)
```

Ex 6.6. Explain why this function can't use memoization:

```
def pick_at_random(seq) :
    return seq[random.randint(0, len(seq)-1)]
```

Ex. 6.7. Explain why this function can't use memoization:

```
f = open('data', 'r')

def next_n_lines(n) :
    lines = ''
    for i in range(n) :
        lines += f.readline()
    return lines
```

0-1 Knapsack.

Given a capacity c and the value and weight of n items in arrays V and W, find a subset of the n items whose total weight is less than or equal to the capacity and whose total value is maximal.

V	20	15	90	100
W	1	2	4	5
	0	1	2	3

$c = 7$

set	weight	value	
$\{2, 3\}$	9	190	*exceeds capacity*
$\{1, 3\}$	7	115	*not optimal*
$\{0, 1, 2\}$	7	125	*optimal*

Knapsack

Let $B[i][j]$ be the value of the best way to fill remaining knapsack capacity i using only items 0 through j. Then $B[c][n-1]$ is the value-solution to the entire problem, that is,

$$B[c][n-1] = \max_K \sum_{j=0}^{n-1} K[j]V[j]$$

In the general case we have the choice between

$$\underbrace{V[j]}_{\substack{\text{value of}\\ \text{the } j\text{th}\\ \text{item}}} \quad \underbrace{+ B[\underbrace{i - W[j]}_{\substack{\text{remaining}\\ \text{capacity}\\ \text{after}\\ \text{taking the}\\ j\text{th item}}}][j-1]}_{\substack{\text{The best way to}\\ \text{fill the remaining}\\ \text{capacity with the}\\ \text{remaining items}}} \qquad \text{versus} \qquad \underbrace{B[i][j-1]}_{\substack{\text{The best way to}\\ \text{fill the unchanged}\\ \text{capacity with the}\\ \text{remaing items}}}$$

Knapsack

$$B[i][j] = \begin{cases} 0 & \text{if } j = 0 \text{ and } W[0] > i \quad \text{(0th doesn't fit)} \\ V[0] & \text{if } j = 0 \text{ and } W[0] \leq i \quad \text{(0th fits)} \\ B[i][j-1] & \text{if } W[j] > i \quad \text{(}j\text{th doesn't fit)} \\ \max\left\{ \begin{array}{l} V[j] + B[i - W[j]][j-1], \\ B[i][j-1] \end{array} \right\} & \text{otherwise} \quad \text{(}j\text{ fits)} \end{cases}$$

Problem:

item	0	1	2	3	4
weight	1	11	6	5	4
value	150	990	70	50	40

4	0/S	150/S	150/S	150/S	150/S	190/T	200/S	220/S	220/S	220/S	240/T
3	0/S	150/S	150/S	150/S	150/S	150/S	200/T	220/S	220/S	220/S	220/S
2	0/S	150/S	150/S	150/S	150/S	150/S	150/S	220/T	220/T	220/T	220/T
1	0/S	150/S	150/S	150/S	150/S	150/S	150/S	150/S	150/S	150/S	150/S
0	0/S	150/T	150/T	150/T	150/T	150/T	150/T	150/T	150/T	150/T	150/T
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

capacities

Longest common subsequence.

Given two sequences, find the longest subsequence that they have in common.

D A T A S T R U C T U R E S
A L G O R I T M S

A A A A A B
A B A A A A

not

A A A A A B
A B A A A A

A A A A A B A A A A
A B A A A A

not

A A A A A B A A A A
A B A A A A

Longest common subsequence

Let $L[i][j]$ be the length of the longest common subsequence of $a[:i]$ and $b[:j]$. Then $L[n][m]$ is the top-level problem.

$$L[i][j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 & \text{(At least one prefix is empty)} \\ 1 + L[i-1][j-1] & \text{if } i \neq 0 \text{ and } j \neq 0 & \text{(Last symbols match—take it)} \\ & \text{and } a[i-1] = b[j-1] & \\ \max\{L[i][j-1], & & \text{(Last symbols don't match—} \\ \qquad L[i-1][j]\} & \text{otherwise} & \text{ skip one)} \end{cases}$$

For subsequences `algorithms` and `datastructures`, the table would be:

s	**10**	0	0/1	1/1	2/1	2/1	3/0	3/-1	3/-1	3/-1	3/-1	3/1	3/1	3/1	3/1	4/0
m	**9**	0	0/1	1/1	2/1	2/1	2/1	2/1	2/1	2/1	2/1	3/1	3/1	3/1	3/1	3/1
h	**8**	0	0/1	1/1	2/1	2/1	2/1	2/1	2/1	2/1	2/1	3/1	3/1	3/1	3/1	3/1
t	**7**	0	0/1	1/1	2/0	2/-1	2/-1	2/0	2/1	2/1	2/1	3/0	3/-1	3/-1	3/-1	3/-1
i	**6**	0	0/1	1/1	1/1	1/1	1/1	1/1	2/1	2/1	2/1	2/1	2/1	2/1	2/1	2/1
r	**5**	0	0/1	1/1	1/1	1/1	1/1	1/1	2/0	2/-1	2/-1	2/-1	2/-1	2/0	2/-1	2/-1
o	**4**	0	0/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
g	**3**	0	0/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
l	**2**	0	0/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
a	**1**	0	0/1	1/0	1/-1	1/0	1/-1	1/-1	1/-1	1/-1	1/-1	1/-1	1/-1	1/-1	1/-1	1/-1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
			d	a	t	a	s	t	r	u	c	t	u	r	e	s

Matrix multiplication.

> *Given $n+1$ dimensions of of n matrices to be multiplied, find the optimal order in which to multiply the matrices, that is, find the parenthesization of the matrices that will minimize the number of scalar multiplications.*

Assume the following matrices and dimensions: $A, 3 \times 5$; $B, 5 \times 10$; $C, 10 \times 2$, $D, 2 \times 3$; $E, 3 \times 4$.

$$
\begin{array}{lrcl}
(A \times B) \times (C \times (D \times E)) & 3 \cdot 5 \cdot 10 + 2 \cdot 3 \cdot 4 + 10 \cdot 2 \cdot 4 + 3 \cdot 10 \cdot 4 & = & 374 \\
(A \times (B \times C)) \times (D \times E) & 5 \cdot 10 \cdot 2 + 2 \cdot 3 \cdot 4 + 3 \cdot 5 \cdot 2 + 3 \cdot 2 \cdot 4 & = & 178 \\
A \times (B \times (C \times (D \times E))) & 2 \cdot 3 \cdot 4 + 10 \cdot 2 \cdot 4 + 5 \cdot 10 \cdot 4 + 3 \cdot 5 \cdot 4 & = & 364
\end{array}
$$

Matrix multiplication.

$$\begin{pmatrix}2 & 8\\ 5 & 7\end{pmatrix}\begin{pmatrix}3 & 6\\ 1 & 4\end{pmatrix} = \begin{pmatrix}2\cdot 3+8\cdot 1 & 2\cdot 6+8\cdot 4\\ 5\cdot 3+7\cdot 1 & 5\cdot 6+7\cdot 4\end{pmatrix} = \begin{pmatrix}14 & 44\\ 22 & 58\end{pmatrix}$$

$$\begin{pmatrix}1 & 3 & 12\\ 2 & 7 & 11\end{pmatrix}\begin{pmatrix}4 & 10\\ 8 & 6\\ 9 & 5\end{pmatrix} = \begin{pmatrix}1\cdot 4+3\cdot 8+12\cdot 9 & 1\cdot 10+3\cdot 6+12\cdot 5\\ 2\cdot 4+7\cdot 8+11\cdot 9 & 2\cdot 10+7\cdot 6+11\cdot 5\end{pmatrix} = \begin{pmatrix}136 & 88\\ 163 & 117\end{pmatrix}$$

$$\begin{pmatrix}1 & 2 & 5\\ 6 & 8 & 9\end{pmatrix}\begin{pmatrix}3\\ 7\\ 4\end{pmatrix} = \begin{pmatrix}1\cdot 3+2\cdot 7+5\cdot 4\\ 6\cdot 3+8\cdot 7+9\cdot 4\end{pmatrix} = \begin{pmatrix}37\\ 110\end{pmatrix}$$

Matrix multiplication

Let $M[i][j]$ be the least number of scalar multiplications needed to multiply submatrices A_i through A_j, inclusive. Then $M[0][n-1]$ is the top-level problem.

$$M[i][j] = \begin{cases} 0 & \text{if } i = j \quad \text{(Only one matrix)} \\ \min_{i \le k < j}\{M[i][k] + D[i]D[k+1]D[j+1] + M[k+1][j]\} & \text{otherwise} \quad \text{(Find the best way to cut this series)} \end{cases}$$

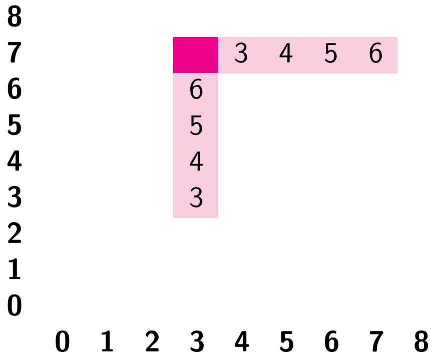
Close-up of the recursive case:

$$\underbrace{M[i][k]}_{\substack{\text{minimum} \\ \text{multiplica-} \\ \text{tions for} \\ (A_i \cdots A_k)}} + \underbrace{D[i]D[k+1]D[j+1]}_{\substack{\text{multiplications for} \\ (A_i \cdots A_k)(A_{k+1} \cdots A_j)}} + \underbrace{M[k+1][j]}_{\substack{\text{minimum} \\ \text{multiplica-} \\ \text{tions for} \\ (A_{k+1} \cdots A_j)}}$$

Which subproblems does subproblem $M[3][7]$ depend on?

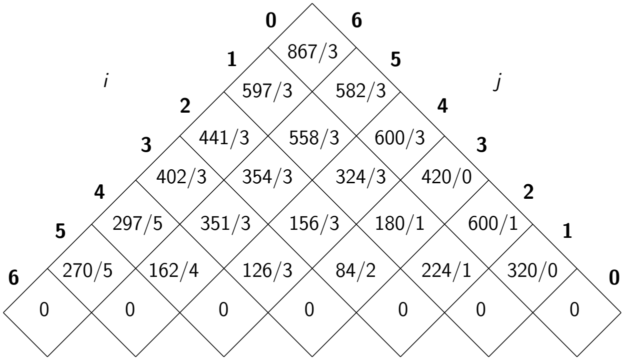
For k ranging over $[3, 7)$:

k	3	4	5	6
subproblems	$(M[3][3], M[4][7])$	$(M[3][4], M[5][7])$	$(M[3][5], M[6][7])$	$(M[3][6], M[7][7])$



$$M[i][j] = \begin{cases} 0 & \text{if } i = j \quad \text{(Only one matrix)} \\ \min_{i \le k < j}\{M[i][k] + \\ \qquad D[i]D[k+1]D[j+1] + \\ \qquad M[k+1][j]\} & \text{otherwise} \quad \text{(Find the best way to cut this series)} \end{cases}$$

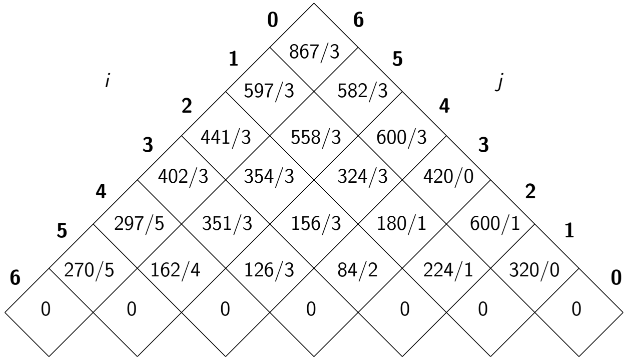
Problem: $D = [10, 8, 4, 7, 3, 6, 9, 5]$



i \ j	0	1	2	3	4	5	6
0	0	320/0	600/1	420/0	600/3	582/3	867/3
1		0	224/1	180/1	324/3	558/3	597/3
2			0	84/2	156/3	354/3	441/3
3				0	126/3	351/3	402/3
4					0	162/4	297/5
5						0	270/5
6							0

	Cell indices (i,j)
Base cases	(0,0), (1,1), (2,2), (3,3), (4,4), (5,5), (6,6)
	(0,1), (1,2), (2,3), (3,4), (4,5), (5,6)
	(0,2), (1,3), (2,4), (3,5), (4,6)
	(0,3), (1,4), (2,5), (3,6)
	(0,4), (1,5), (2,6)
	(0,5), (1,6)
Top-level problem	(0,6)

Problem: $D = [10, 8, 4, 7, 3, 6, 9, 5]$



Coming up:

Catch up on projects. . .
(Recommended: Do **LL RB** *project for your own practice)*

Due **Thurs, Apr 3**
Read Section 6.4
Take quiz (DP algorithms)

(See Canvas for practice problems for Test 2)