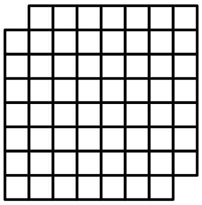
CS4102 Algorithms
Spring 2019

Can you fill a 8×8 board with the corners missing using dominoes?

Can you tile this?



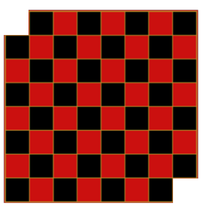
With these?

1

CS4102 Algorithms
Spring 2019

Can you fill a 8×8 board with the corners missing using dominoes?

Can you tile this?



With these?

2

Today's Keywords

- Midterm Review
- Dynamic Programming
- Longest Common Subsequence

4

CLRS Readings

- Chapter 15

5

Administrativa

- HW5 due March 27 at 11pm
  - Seam Carving!
  - Dynamic Programming (implementation)
  - Java or Python
- Grades Released
  - HW3
  - Midterm

6

Dynamic Programming

- Requires Optimal Substructure
  – Solution to larger problem contains the solutions to smaller ones
- Idea:
  1. Identify recursive structure of the problem
     - What is the "last thing" done?
  2. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest
  3. Save solution to each subproblem in memory

7

Generic Top-Down Dynamic Programming Soln

```
mem = {}
def myDPalgo(problem):
        if mem[problem] not blank:
                return mem[problem]
        if baseCase(problem):
                solution = solve(problem)
                mem[problem] = solution
                return solution
        for subproblem of problem:
                subsolutions.append(myDPalgo(subproblem))
        solution = OptimalSubstructure(subsolutions)
        mem[problem] = solution
        return solution
```

8

Longest Common Subsequence

Given two sequences X and Y, find the length of their longest common subsequence

Example:
$X = A\underline{T}\underline{C}\underline{T}G\underline{A}T$
$Y = TGCATA$
$LCS = TCTA$

Brute force: Compare every subsequence of X with Y
$\Omega(2^n)$

9

Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
- Idea:
  1. Identify recursive structure of the problem
     - What is the "last thing" done?
  2. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest
  3. Save solution to each subproblem in memory

10

1. Identify Recursive Structure

Let $LCS(i,j)$ = length of the LCS for the first i characters of X, first j character of Y

Find $LCS(i,j)$:

Case 1: $X[i] = Y[j]$

$X = ATCTGCGT$
$Y = TGCATAT$
$LCS(i,j) = LCS(i-1, j-1) + 1$

Case 2: $X[i] \neq Y[j]$

$X = ATCTGCGA$
$Y = TGCATAT$
$LCS(i,j) = LCS(i, j-1)$

$X = ATCTGCGT$
$Y = TGCATAC$
$LCS(i,j) = LCS(i-1, j)$

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

11

Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
- Idea:
  1. Identify recursive structure of the problem
     - What is the "last thing" done?
  2. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest
  3. Save solution to each subproblem in memory

12

1. Identify Recursive Structure

Let $LCS(i,j) =$ length of the LCS for the first i characters of X, first j character of Y

Find $LCS(i,j)$:

Case 1: $X[i] = Y[j]$

$X = ATCTGCGT$
$Y = TGCATAT$
$LCS(i,j) = LCS(i-1,j-1) + 1$

Case 2: $X[i] \neq Y[j]$

$X = ATCTGCGA$
$Y = TGCATAT$
$LCS(i,j) = LCS(i,j-1)$

$X = ATCTGCGT$
$Y = TGCATAC$
$LCS(i,j) = LCS(i-1,j)$

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1,j-1)+1 & \text{if } X[i] = Y[j] \\ \max(LCS(i,j-1), LCS(i-1,j)) & \text{otherwise} \end{cases}$$

Read from M[i,j] if present

Save to M[i,j]

Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
- Idea:
  1. Identify recursive structure of the problem
     - What is the "last thing" done?
  2. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest
  3. Save solution to each subproblem in memory

2. Solve in a Good Order

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1,j-1)+1 & \text{if } X[i] = Y[j] \\ \max(LCS(i,j-1), LCS(i-1,j)) & \text{otherwise} \end{cases}$$

$Y =$ \ $X =$		0	A 1	T 2	C 3	T 4	G 5	A 6	T 7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

To fill in cell (i,j) we need cells $(i-1,j-1), (i-1,j), (i,j-1)$
Fill from Top->Bottom, Left->Right (with any preference)

Run Time?

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

$Y =$ \ $X =$		0	A 1	T 2	C 3	T 4	G 5	A 6	T 7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Run Time: $\Theta(n \cdot m)$ (for $|X| = n$, $|Y| = m$)

16

Reconstructing the LCS

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

$Y =$ \ $X =$		0	A 1	T 2	C 3	T 4	G 5	A 6	T 7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Start from bottom right,
if symbols matched, print that symbol then go diagonally
else go to largest adjacent

17

Reconstructing the LCS

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

$Y =$ \ $X =$		0	A 1	T 2	C 3	T 4	G 5	A 6	T 7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Start from bottom right,
if symbols matched, print that symbol then go diagonally
else go to largest adjacent

18

Reconstructing the LCS

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

$Y =$ \ $X =$		0	A 1	T 2	C 3	T 4	G 5	A 6	T 7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Start from bottom right,
if symbols matched, print that symbol then go diagonally
else go to largest adjacent

19