6.006 Introduction to Algorithms
Spring 2008

6.006 Recitation

Build 2008.36

6.006 Proudly Presents

- PS 6
- Super Mario Brothers
- Points Back on Tests
- DP vs. Minimum-Cost Paths

PS 6 Out

- The best way to gauge your understanding of Dynamic Programming
- Do fib (fibonacci) over the weekend
  - Come get help if you can't do it quickly!
- Do the other problems as soon as you understand them

Beating Super Mario: The Vision

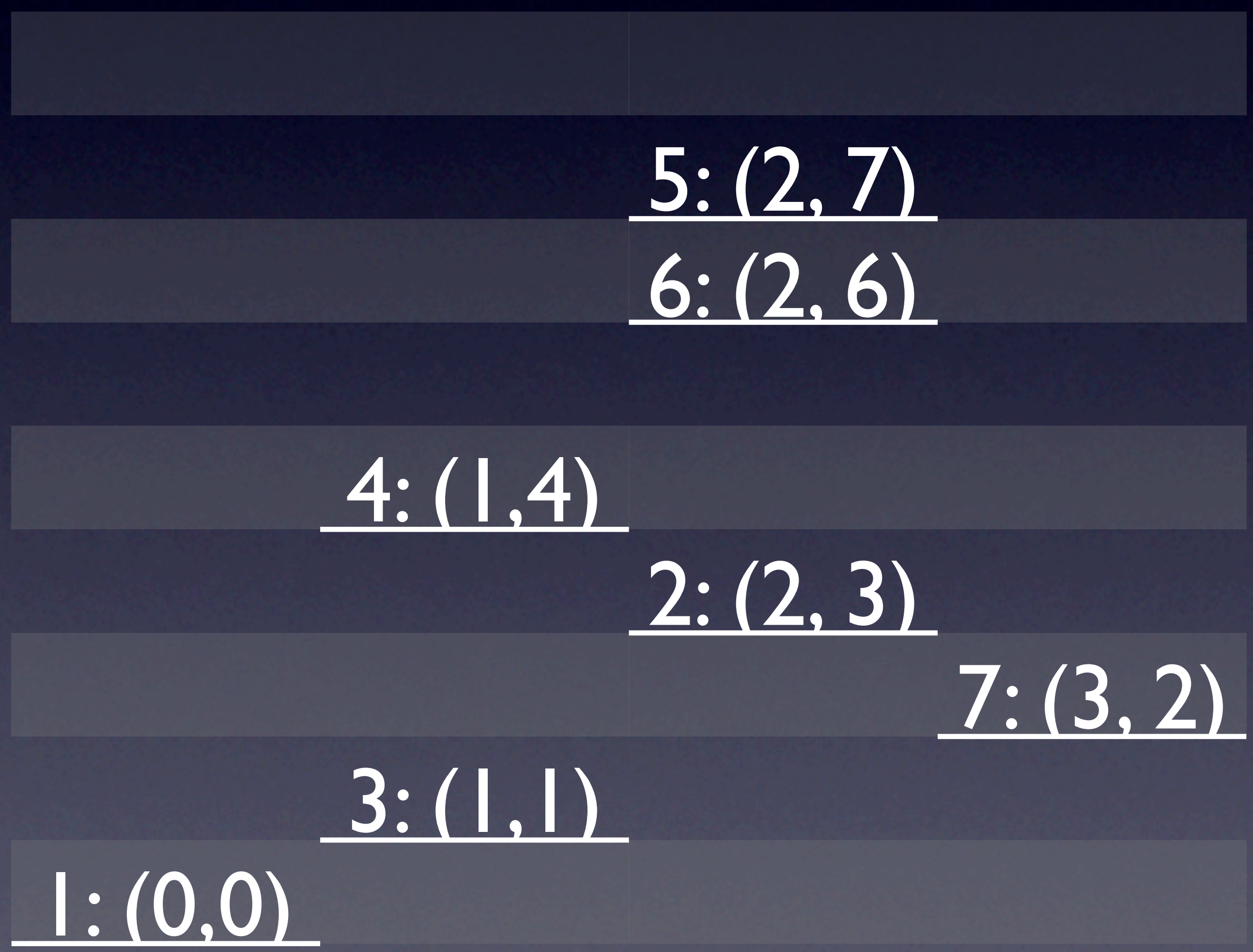
1. Abstract into 6.006 problem

2. Solve using DP

3. pwn

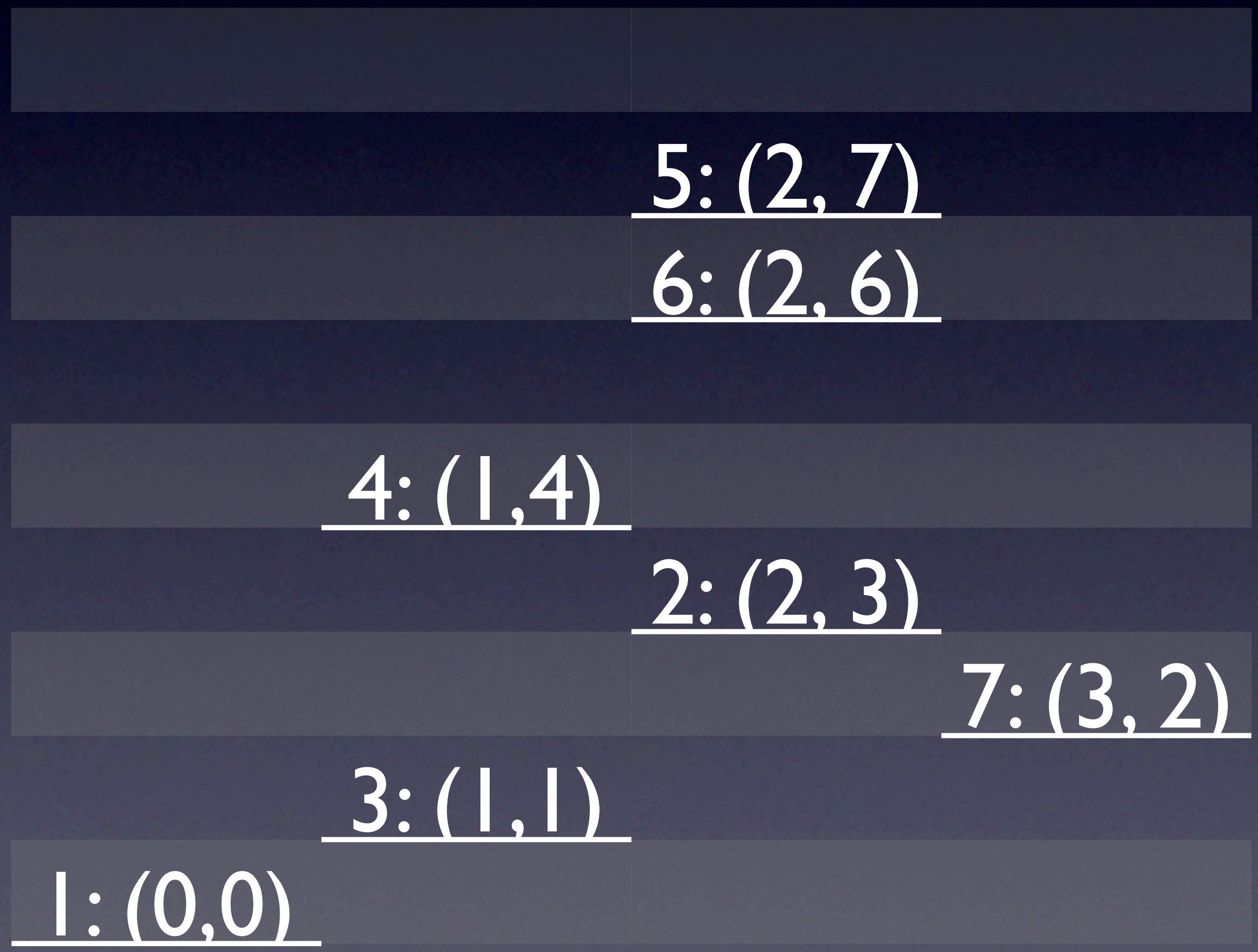
Platforming I

- P platforms, at (x_i, y_i)
- Starting on platform 1, want to get to platform P
- Pure pwnage
  - Always move right
  - Minimum # moves



Platforming II

- Moves from (x, y)
  - walk: (x+1, y)
  - jump: (x+1, y+1) or (x+1, y+2)
  - super-jump: (x+1, y+3) or (x+1, y+4)
  - fall: (x+d, y-d-d’) as long as d+d’ < 5



Platforming: Solution 1

- Problem: the minimum number of moves from platform 1 to platform P
- Optimal sub-structure
  - assume the optimal solution stops at platform Q right before moving to P
  - then the optimal solution must get from platform 1 to Q w/ the min. no. of moves

Platforming: Solution II

- d[p] = minimum # of moves to get to p
- parent_p[p] = parent platform for p
- parent_m[p] = parent move for p
- bottom-up solution: sort the platforms by their x coordinate, then d[p] only depends on d[p'] where p' < p

Platforming: Running Time

- Subproblems
  - one per platform - P in total
- Time per subproblem
  - looking back at previous platforms - $O(P)$
- Total running time - $O(P^2)$

Points Back on Tests

- Multiple-choice test (think SATs)
- Each answer is an alphabet letter (for SAT, the alphabet is A-E)
- Single correct answer for each question

Qtn.	Your Ans.	Correct
1	A	A
2	B	C
3	A	B
4	C	A
5	D	C
6	A	D
7	E	A
8	E	E

Points Back on Tests II

- Step 1: Claim that you made an error when transcribing answers
- Step 2: Hire a damn good lawyer, claim that you did multiple mistakes
- Outcome: Longest Common Subsequence

Qtn.	Your Ans.	Correct
1	**A**	**A**
2	**B**	C
3	**A**	**B**
4	**C**	**A**
5	**D**	**C**
6	**A**	**D**
7	**E**	**A**
8	E	**E**

Points Back on Tests: Towards a Solution

- $x = [A, B, A, C, D, A, E]$
- $y = [A, C, B, A, A, B, E]$
- Solution: a list of pairs (s_i, t_i) s.t.
  - $x[s_i] = y[t_i]$
  - $s_i < s_j$ and $t_i < t_j$ for any $_i < _j$

Points Back on Tests: Solution 1

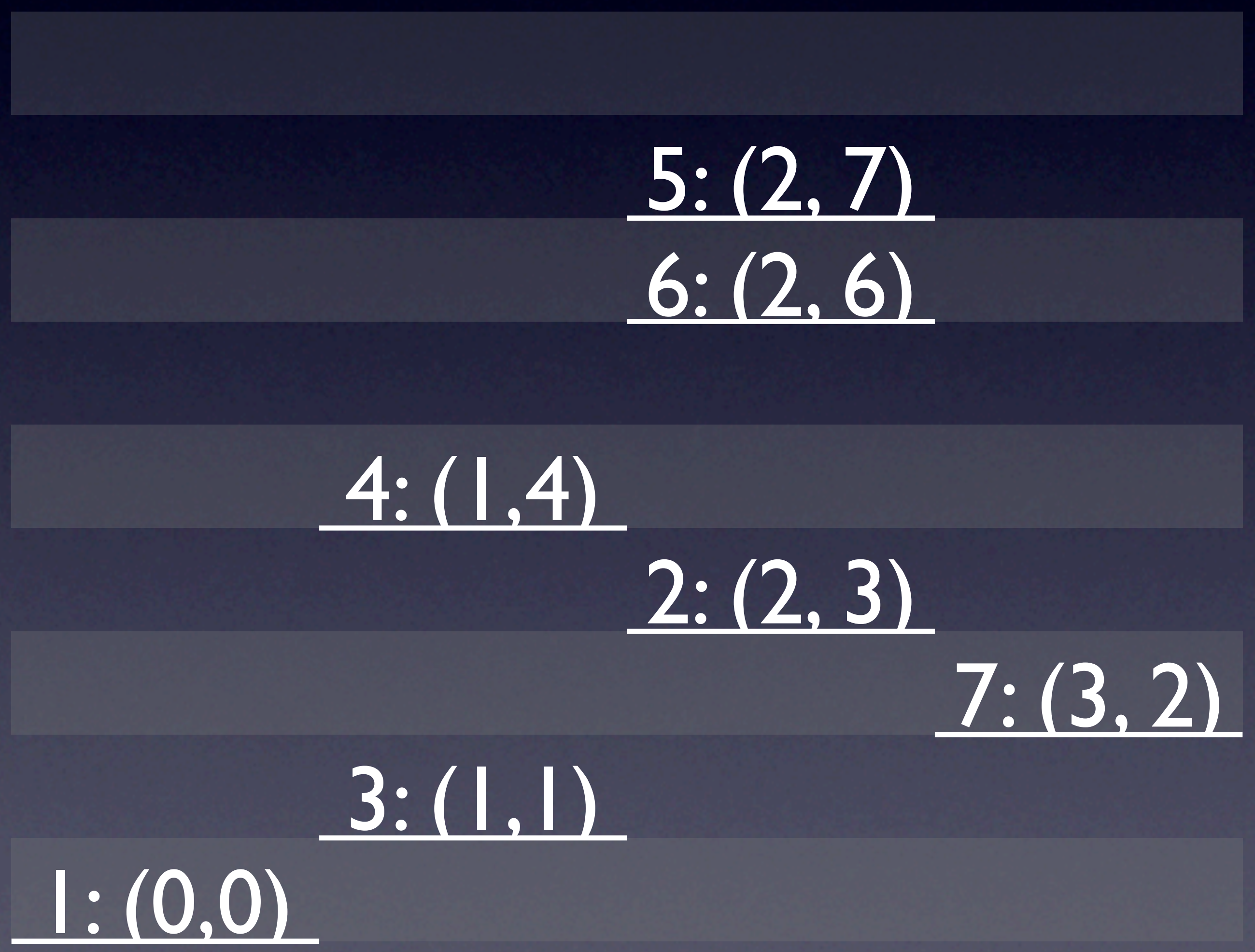
- Want: the longest common sequence in x, y
- Optimal sub-structure:
  - assume answer (s_1, t_1) ... $(s_{n-1}, t_{n-1}), (s_n, t_n)$
  - then (s_1, t_1) ... (s_{n-1}, t_{n-1}) must be the longest common sequence of $x[1: s_{n-1}]$ and $x[1: t_{n-1}]$

Points Back on Tests: Solution II

- d[i][j] = len. of max. common sequence of x[1:i] and y[1:j]
- d[0][j] = 0, d[i][0] = 0
- d[i][j] only depends on d[i-1][j-1], d[i-1][j], and d[i][j-1], so we can build d bottom-up for i from 0 to len(x) and for j from 0 to len(y)

DP vs. Min-Cost Paths: Platforming I

- Each platform is a node
- A move between P and Q is a directed edge (P, Q) of cost 1
- Want: min-cost path between node 1 and P
- Parents in DP: same as the parents in single-source min-cost paths



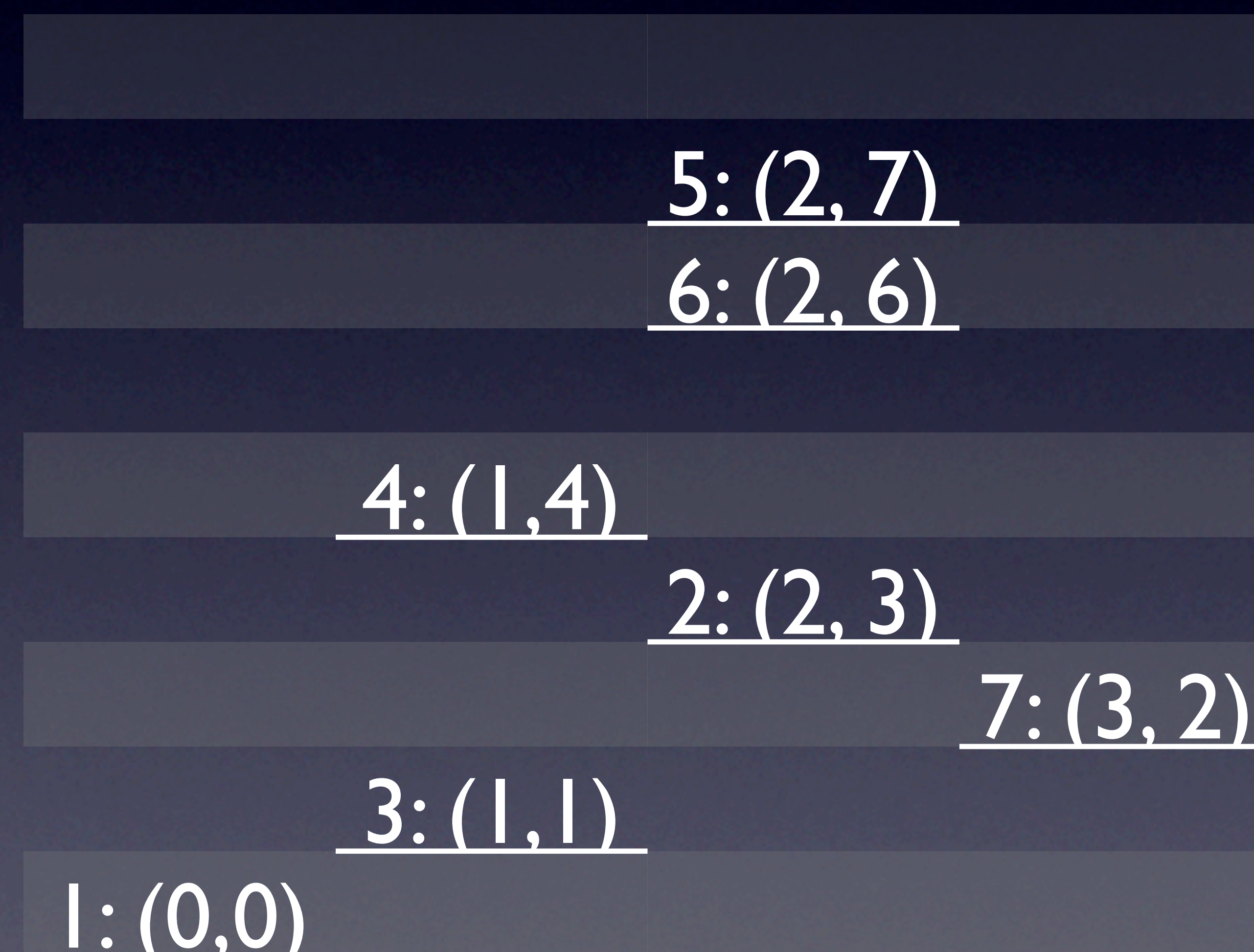
DP vs. Min-Cost Paths: Platforming II

- We only move right ⇒

  all edges are from left to right ⇒ sorting by x

  computes a topologic ordering

- Bottom-up DP is the same as computing single-source min-cost paths in a DAG



DP vs. Min-Cost Paths: Points Back on Tests 1

- subproblems: d[i][j] ⇒ a node is a tuple (i, j)
- 0-weight edges from (i,j) to (i,j+1) and to (i+1,j)
- Edge (i,j) to (i+1,j+1) has weight 1 if x[i] = y[j]
- Want: max-cost path from (0, 0) to (|x|, |y|)

Qtn.	Your Ans.	Correct
1	A	A
2	B	C
3	A	B
4	C	A
5	D	C
6	A	D
7	E	A
8	E	E

DP vs. Min-Cost Paths: Points Back on Tests I

- Edges are (i,j) to (i+1, j), (i+1, j+1) and (i, j+1) ⇒

  lexicographical ordering is a topological order

- So we can do min-cost path in DAGs by multiplying edges by -1

- DP does exactly that!

Qtn.	Your Ans.	Correct
1	**A**	**A**
2	**B**	C
3	**A**	**B**
4	**C**	**A**
5	**D**	**C**
6	**A**	**D**
7	**E**	**A**
8	E	**E**