# Dynamic Programming (II)

Anson Ho

# Contents

- DAG DP
- Dimension reduction
  - Rolling DP
- Tree DP
- Bitwise DP

# What is DP

- State(s)
- Transition formula
- Base case(s)

- Memorization

- Time complexity
- Space complexity

# DAG

- Directed acyclic graph
- Visualization of transition

# DAG

- Fibonacci number
- Number of ways to pay $10
- Calculate $nCr = \frac{n!}{r!(n-r)!}$
  - O(N) vs O($N^2$)
- Shortest path

# DAG

- Input a list of bus routes with time intervals
- Given that:
  - all routes are one-way
  - impossible to visit the same bus stop after get on a bus
- Output the length of the longest bus journey

# DAG

- Note that the graph is a DAG
  - but not necessarily connected
- Longest path in DAG
- Find topological order
  - Graph (II)

# DAG

- All DP problems can be represented by DAG?
- It depends on definition
- DAG of possible transitions or DAG of used transitions

# DAG

- Given a necklace with number written on each beads
- Form a chain by destroying a bead
- Output all possible products of numbers on the chain

# DAG

- Possible approach
  - x[1] * x[2] * … * x[n] / x[i]
- DP approach
  - dp[i] = dp[i – 1] * x[i – 1] / x[i]
    - chain or cycle?
- Correct approach
  - partial “sum”
    - Mathematics in OI (I)

# Dimension reduction

- Rolling DP

- Reduce dimension for space
- Not for time
  - in DP (III)

# Dimension reduction

- Number of paths in grids
  - From top-left to bottom right

- 0: Only ↓ and → are allowed
- 1: Some blocked grids
- 2: 5000x5000 grid
  5000 blocked  1MB memory

# Dimension reduction

- dp[i][j]
  = f(dp[i-1][j], dp[i-2][j+1])
- Only keep the last two dp[i]

- Avoid moving a large chunk of memory

- i → i mod 3
- i-1 → i-1 mod 3
- i-2 → i-2 mod 3

# Dimension reduction

- Disadvantage
  - cannot do backtracking

# Dimension reduction

- Number of paths in grids
  - From top-left to bottom right

- 0: Only ↓(D) and →(R) are allowed
- 1: Some blocked grids
- 2: 1000x1000 grid
- 3: output kth lexicographically smallest path

# Take a break

and think for the solution

# Tree DP

- What is tree?

- A tree is a DAG
- (A DAG is not necessarily a tree)

- Rooted or not?

# Tree DP

- Rooted
  - parent → child or
    parent ← child
- Height/depth of each node
- Size of each subtree
- Minimax in a game tree

# Tree DP

- Not rooted
  - bidirectional edges
- Number of paths passing through each node
- Sum of path weights
  - path weight = sum of edge weights
  - path weight = product of edge weights

# Tree DP

- Number of “k-subtrees” in unrooted tree
- “k-subtree” means a subgraph which is a tree with k nodes

# Tree DP

- Some of the nodes in a tree are coloured
- For every node, find the number of paths with coloured ends passing through it

# (Extra)

- Every node in a tree is coloured

- Queries with a specific node and a specific colour
- Find the number of paths with ends in that colour passing through that node

# (Extra)

- Given two rooted tree
- Find the minimum number of additional nodes to make the two trees “isomorphic”
- Assignment problem

# Tree DP

- More…
- T141 Bytefest

# Bitmask DP

- State
- dp[133][2] → dp[$101100101_2$]

- Bit manipulation
  - bitwise and (&)
  - bitwise or (|)
  - exclusive or (^)

# Bitmask DP

- Assignment problem
- Matching

- Maximum matching
- Matching with minimum cost

- Polynomial time algorithm exists

# Bitmask DP

- Hamiltonian path
- Graph traversal with visiting each node exactly once
- Count the number of such paths with longest length

# Bitmask DP

- N (<=15) light bulbs
- K (<=30) buttons
  - each control a set of light bulbs

- Find the minimum number of toggling to achieve a specific configuration

# Bitmask DP

- Number of ways to fill a NxM grid with 1x2 (or 2x1) rectangle
  - fully filled
  - no overlapping

- N=2, M=2
- Ans = 2

# Bitmask DP

- CERC14 E Can't stop playing
- http://codeforces.com/gym/100543
- 1D 2048 game
- Given a sequence of $2^i$
- Append each number to left/right in the given order
- Construct a left-right sequence such that everything can be merged into a single cell finally

Thank you