

DP(II)

Theo

Recall

Topics in DP(I)

what is the definition of “State” and “Transition formula”?

Topics

- Memory optimization: Rolling array
- Bitwise DP
- Tree DP

Rolling array

Recall the grid counting problem:

Given a $N \times M$ grid, find out the number of ways to move from $(1,1)$ to (N,M) with right and down movement only.

Rolling array

$f[i][j]$: way to go from $(1,1)$ to (i, j) .

$$f[1][1] = 1$$

$$f[i][j] = \begin{cases} f[i-1][j] + f[i][j-1] & \text{if } (a[i][j] \text{ is passable}) \\ 0 & \text{otherwise} \end{cases}$$

Memory complexitiy? $O(NM)$

Rolling array

if we have $F[1..N][i]$, then we can have $F[1..N][i+1]$ without referring to $F[1..N][i-1]$!

--> holds two column of "F" values is enough

$O(NM)$ --> $O(M)$

Rolling array

Even better --> if $(M > N)$ swap N and M
 $O(M)$ --> $O(\min(N, M))$

Bitwise DP

Consider following problem:

Given a $N \times M$ grid, In how many ways can we place a 1×2 domino (in horizontal or vertical) so that the grid is completely filled and no domino overlap?

$N \leq 1000$, $M \leq 10$

Bitwise DP

Notice that the placement of a domino on column M is determined by the domino config on column $M-1$ and M only.

--> $dp[a][b]$: number of ways to fill column $1..(a-1)$ such that all the columns $1..(a-2)$ is filled, and the shape of column $(a-1)$ is b .

Bitwise DP

Notice that the placement of a domino on column M is determined by the domino config on column $M-1$ and M only.

--> $dp[a][b]$: number of ways to fill column $1..(a)$ such that all the columns $1..(a-1)$ is filled, and the shape of column a is b .

Bitwise DP

how to “define” a shape?

-->as we concerns about the filled “stae” of the (a-1) column only, we can represent the shape as a binary number!

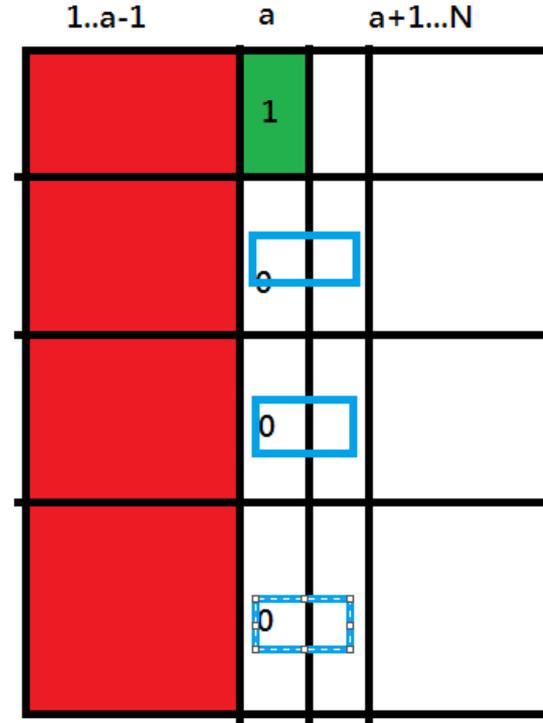
1..a-1	a	a+1..N
	1	
	0	
	0	
	1	

Bitwise DP

Transition?

Given a shape, one can either put all the “0s” with horizontal dominos

--> $dp[i][j]$ contributes to $dp[i + 1][2^M - j]$

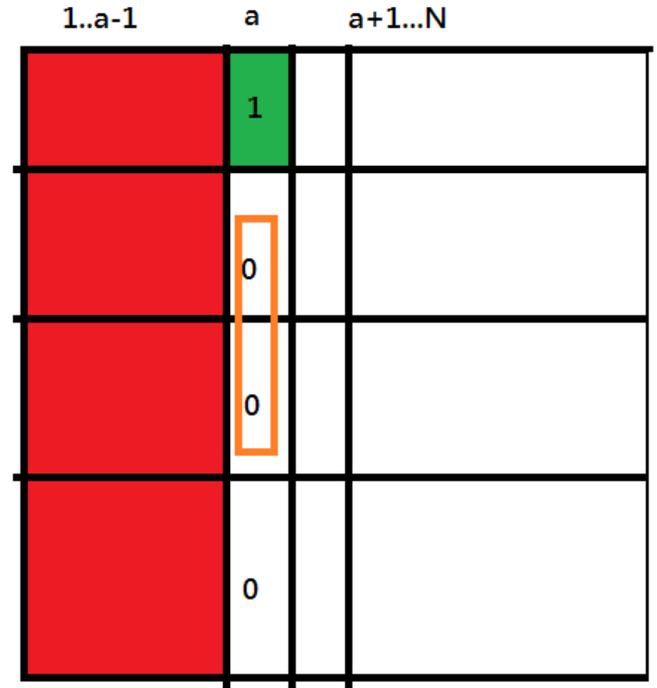


Bitwise DP

or, we can put a vertical domino at the current column.

but after the bottomest "1" slot. Why?

--> $dp[i][j]$ contributes to $dp[i][j+2^k+2^{(k+1)}]$ as long as they are empty



Bitwise DP

Time Complexity?

Number of state = $O(2^M * N)$

Transition per state = $O(M)$

--> $O(MN * 2^M)$

Tree DP

Tree: A graph so that there is only a single path from a vertex to any other vertex.

Many optimization problems in tree can be solved by tree DP.

Not a tree? --> biconnected component it sometimes may work. (Graph III)

Tree DP

Two types of tree:

Rooted: easy

Unrooted: annoying

Tree DP

Assume tree is rooted somewhere.

DP problem in tree --> uses nodes as state,
and states' children as transition formula
reference

Tree DP

Easy example: Given a rooted tree of size N , each vertex is labeled a number, given Q queries, for each query find out the vertex that has the greatest number in some subtree.

Tree DP

Let $F(x)$ be the answer to the problem with subtree x .

$$F(x) = \max(V_x, F(u): u \text{ is } x\text{'s children})$$

Tree DP

Time complexity?

Each vertex will send a $O(1)$ request to its children.

--> time complexity = $O(\text{sum of number of children in the tree}) = O(N)$

Tree DP

Unrooted tree cases are more difficult but sometimes we can translate the unrooted case into rooted case (by assigning a random root to the tree and prove that optimal substructure exist)

Tree DP

A little more difficult example:

Given a (unrooted) weighted tree of size N , find a set of paths so that no two paths uses a same vertex and the sum of weight of paths is maximal.

Tree DP

Rephrase: Given the tree, find the set of edges so that a vertex is connected to at most 2 of the edges in the set and the sum of weight is maximal.

Tree DP

If we root the tree (somewhere), then the answer of a subtree depends on the root, as well as the children of the root, and the number of edge the root can use.

$F(u, x)$: maximal answer of subtree rooted at u and u can connect to at most $x(1..2)$ vertex.

Tree DP

How to calculate $F(u, 1)$?

if we fixed where u connects, let it be v , then

$$F(u, 1) = \text{sum}(F(t, 2): t \text{ is } u\text{'s children} + F(v, 1) - F(v, 2) + w(u, v))$$

red part is fixed!

find a v such that the blue part is maximal.

Tree DP

How to calculate $F(u, 2)$?

Similarly, if u connects to v and y , then

$$F(u, 2) = \text{sum}(F(t, 2): t \text{ is } u\text{'s children} + F(v, 1) - F(v, 2) + w(u, v) + F(y, 1) - F(y, 2) + w(u, y))$$

find maximal and second maximal!

Tree DP

Time complexity:

Number of state = $O(N)$

Each state is “referred” once, therefore sum of transition = $O(N)$

find maximal and second maximal --> $O(1)$

-->total = $O(N)$

Tree DP

Sometimes story are not that happy.
We can try all vertex as root.

Drawback: Complexity increased by a factor of $O(N)$

Tree DP

Problem: Given a tree, Let $V(u)$ is the maximal distance between u and another vertex w . We are to find the minimal $V(u)$ among all u .

Subproblem: How to find $V(x)$ for a fixed x ?

Tree DP

Let $F(a, x)$ be the maximal distance from x to one of its leafs in x 's subtree (relative to a)

$$F(a, x) = \max(F(a, t) + w(x, t) : t \text{ is } x\text{'s children})$$

$$V(a) = F(a, a)$$

Tree DP

Back to main problem

Naive calculate $V(u)$ for all u is $O(N^2)$

calculation of one value of $V(u)$ is $O(N)$.

But when we are doing $V(t)$, perhaps we can save some information from the previous calculations

Tree DP

Transition of root:

First we query the V value for a random root, say u

If we fix one of the u 's neighbour as new root, say v , what do we have beforehand?

Tree DP

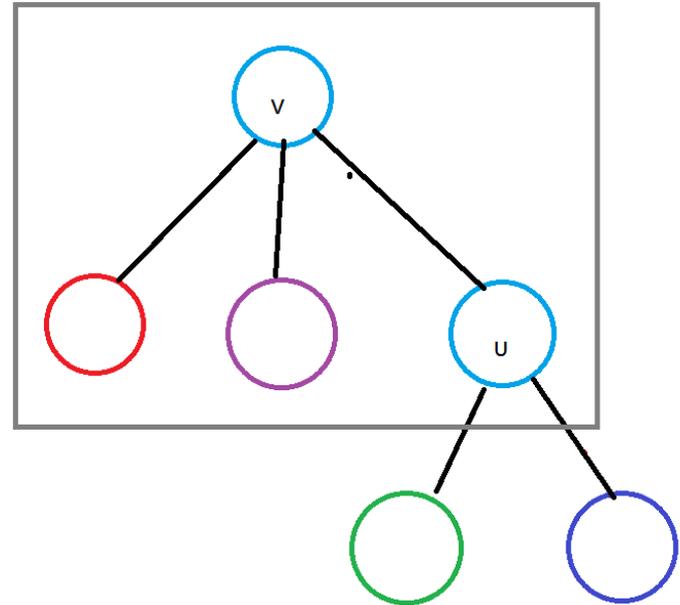
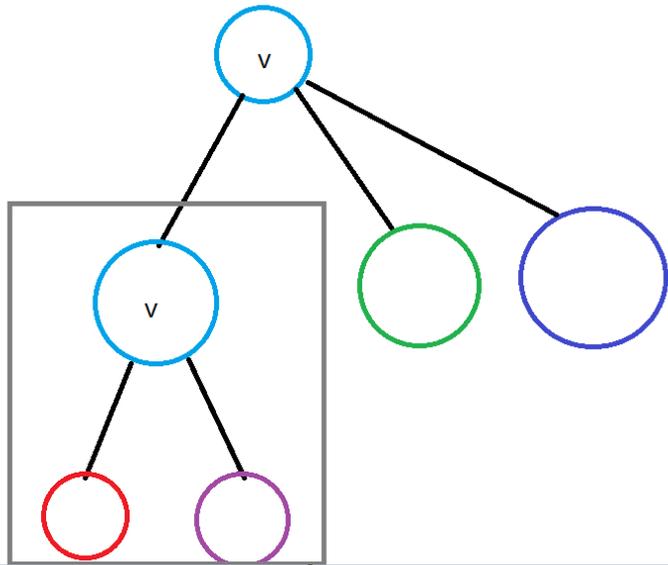
$F(u, x) = \max(F(u, t) + w(x, t): t \text{ is } x\text{'s children})$

$F(u, v) = \max(F(u, t) + w(v, t): t \text{ is } v\text{'s children})$

$F(v, v) = \max(F(v, t) + w(v, t): t \text{ is } v\text{'s children})$

Tree DP

They turn out to be very similar...



Tree DP

$F(u, v) = \max(F(u, t) + w(v, t): t \text{ is } v\text{'s children})$

$F(v, v) = \max(F(u, v), F(v, u) + w(v, u))$

How to calculate red terms in $O(1)$?

Divide into two cases:

Tree DP

- $F(u, u)$ does not depend on v
--> $F(v, u) = F(u, u)$
- $F(u, u)$ depends on v
--> $F(v, u) = \text{second_maximal_of_}F(u, u)$
as we are forbidden to use (u, v)

path

in calculation of $F(v, u)$

--> rotation of roots can be done in $O(1)$

Tree DP

Challenge:

Given a unrooted tree of size N , each vertex is labeled a number, given Q queries, for each query find out the vertex that has the greatest number in some subtree **relative to another neighbour vertex**.