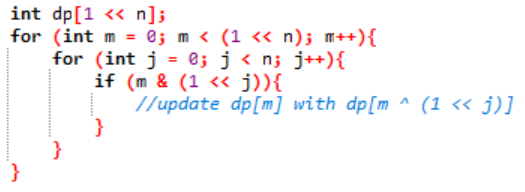
­­­Bitmask DP

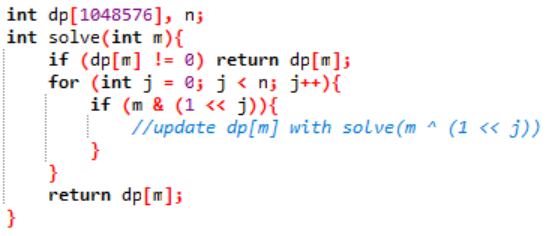
Bitmask DP: dynamic programming but one of the dimensions is bitmask

the bitmask usually represents a state over 0-1 things, or a subset of things

01010 -> 2nd and 4th element exist

Usually something like this:

or this:



Note that this is not a fixed format, you can add dimensions and change the transition in various ways, so you have to be familiar with bitwise operators.

Job assignment

Given a cost matrix where is the amount of money you need to pay to hire worker for the job, assign each workers to distinct jobs to minimize amount of money.

if , can do next permutation

constraint:

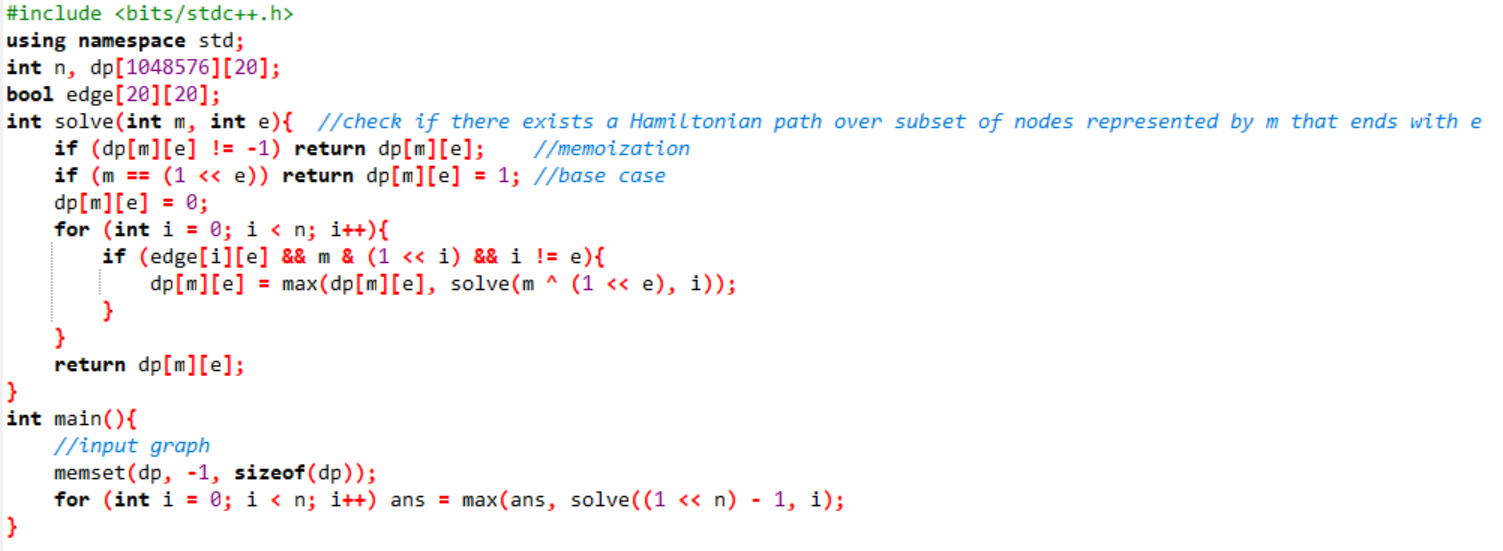
bitmask DP!

dp[3][01011] = min(dp[2][00011], dp[2][01001], dp[2][01010])

Existence of Hamiltonian path

transition:

base case:

time complexity:

Can we do faster??? ??

Actually yes!! State compression!

bit in the new is

so the itself is now a mask, where the bit represent if there exist a Hamiltonian path over that ends in . Don’t get them confused!

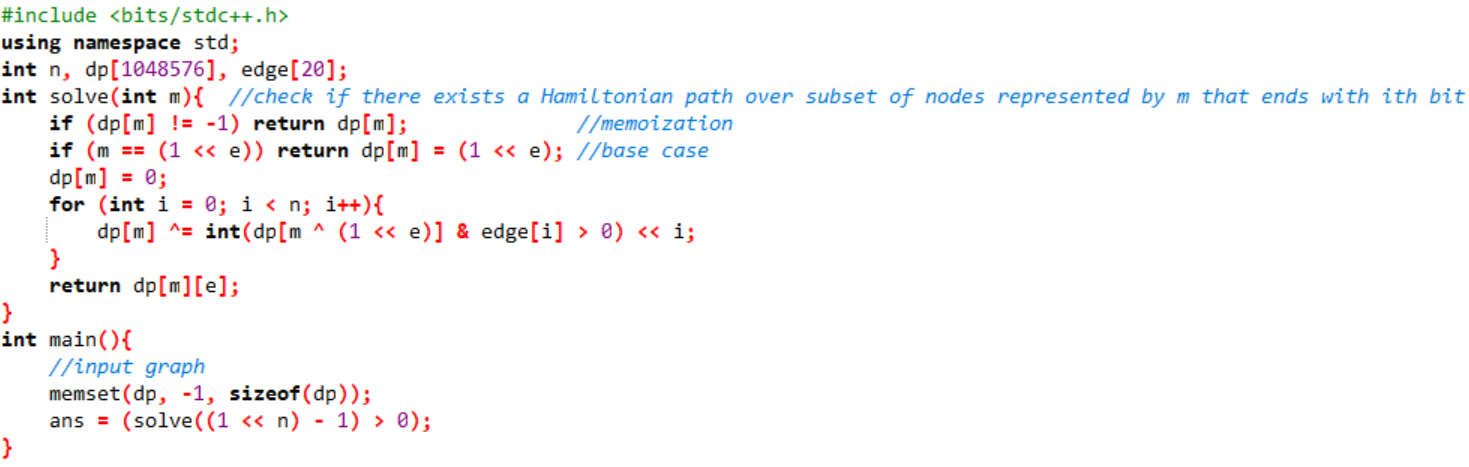
But how about the transition???

Another big brain technique: representing the graph with masks

bool adjacency matrix [n][n] 🡪 int adjacency matrix but masks [n]

Now with these tools provided, try to think of the transition yourself!

Transition:

time: wow!

Sum over subset DP

Given a fixed array of integers, calculate for all the sum of all such that is an subset of .

f(10100) = a[00000]+a[00100]+a[10000]+a[10100]

just iterate submasks of all masks: ?

Actually no!

for each bit, it can be: 0 in mask and 0 in submask, 1 in mask and 0 in submask, or 1 in mask and 1 in submask

Therefore it is . This knowledge is useful when you come up with some transition formula that involves iterating all the submask of all masks.

However, we can do better!

Flaw: is visited by masks

Let .

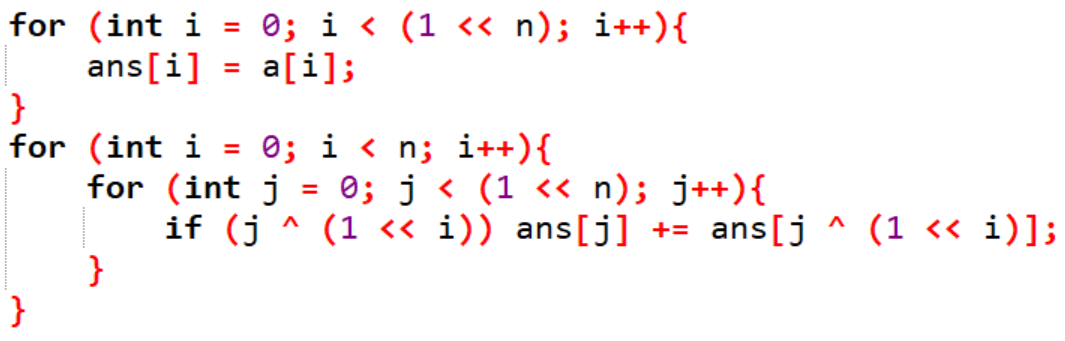
Let

or

For example, dp[101] (101, 100, 001, 000) becomes dp[**10**1][1] (**10**1, **10**0), dp[**1**01][2] (**1**01, **1**00) and dp[101][3] (101, 100, 001, 000) (bold: bits that cannot differ)

Now we can set up a recurrence relation!

If the ith bit is 0 the submasks cannot have a 1 at the ith bit, .

Else, there are two choices: one with 1 at the ith bit () and one with 0 at the ith bit ().

^efficient implementation that reuse the array, it works because j ^ (1 << i) is always smaller than j, which means the answer to that state (ans[j ^ (1 << i] == dp[j ^ (1 << i)][i + 1]) is already calculated when updating the current one (from dp[j][i] to dp[j][i + 1])

This technique can be used to do M1830 lazy tutor

find other bitmask DP problems in dp tag