CP-Algorithms

Page Authors

Search

Dynamic Programming on Broken Profile. Problem "Parquet"

Table of Contents

Common problems solved using DP on broken profile include:

- finding number of ways to fully fill an area (e.g. chessboard/grid) with some figures (e.g. dominoes)
- finding a way to fill an area with minimum number of figures
- finding a partial fill with minimum number of unfilled space (or cells, in case of grid)
- finding a partial fill with the minimum number of figures, such that no more figures can be added

Problem "Parquet"

Problem description. Given a grid of size $N \times M$. Find number of ways to fill the grid with figures of size

2×1 (no cell should be left unfilled, and figures should not overlap each other).

Let the DP state be: `D[i][mask]`, where `i = 1 ... N`, and `mask = 0 ... 2^M-1`. `i` respresents number of rows in the current grid, and `mask` is the state of last row of current grid. If `j`-th bit of `mask` is 0 then the corresponding cell is filled, otherwise it is unfilled. Clearly, the answer to the problem will be `D[N][0]`.

We will be building the DP state by iterating over each `i = 1 ... N` and each `mask = 0 ... 2^M-1`, and for each `mask` we will be only transitioning forward, that is, we will be *adding* figures to the current grid.

Implementation

```
int n, m;
vector < vector<long long> > d;


void calc (int x = 0, int y = 0, int mask = 0,
{
    if (x == n)
        return;
    if (y >= m)
        d[x+1][next_mask] += d[x][mask];
    else
    {
        int my_mask = 1 << y;
        if (mask & my_mask)
            calc (x, y+1, mask, next_mask);
        else
        {
            calc (x, y+1, mask, next_mask | my
            if (y+1 < m && ! (mask & my_mask)
                calc (x, y+2, mask, next_mask)
        }
    }
}


int main()
{
    cin >> n >> m;

    d.resize (n+1, vector<long long> (1<<m));
    d[0][0] = 1;
    for (int x=0; x<n; ++x)
        for (int mask=0; mask<(1<<m); ++mask)
            calc (x, 0, mask, 0);

    cout << d[n][0];

}
```