Last update: September 24, 2023 Translated From: e-maxx.ru

Dynamic Programming on Broken Profile. Problem "Parquet"

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: $dp[i, mask]$, where $i = 1, \dots N$ and $mask = 0, \dots 2^M - 1$.

i represents 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 $dp[N, 0]$.

We will be building the DP state by iterating over each $i = 1, \cdots N$ and each $mask = 0, \dots 2^M - 1$, 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> > dp;


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


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

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

    cout << dp[n][0];

}
```

Practice Problems

- UVA 10359 - Tiling
- UVA 10918 - Tri Tiling
- SPOJ GNY07H (Four Tiling)
- SPOJ M5TILE (Five Tiling)
- SPOJ MNTILE (MxN Tiling)
- SPOJ DOJ1
- SPOJ DOJ2
- SPOJ BTCODE_J
- SPOJ PBOARD
- ACM HDU 4285 - Circuits
- LiveArchive 4608 - Mosaic
- Timus 1519 - Formula 1
- Codeforces Parquet

References

- Blog by EvilBunny
- TopCoder Recipe by "syg96"
- Blogpost by sk765

Contributors:
s9v (52.75%) nzec (36.26%) adamant-pwn (6.59%) jakobkogler (3.3%) therealak12 (1.1%)