Citadel *2022 Sep28*

Given a square matrix (as vector of vector) of integers, pick one number from each row, so that no two numbers lie on the same column to get a maximum sum. Do it once with recursive method, find its complexity, then optimize.

```
// given that the answer of example below is : 863 + 383 + 343 + 959 + 767 = 3315
std::vector<std::vector<int>> A =
{
    {  7,  53, 183, 439, 863},
    {497, 383, 563,  79, 973},
    {287,  63, 343, 169, 583},
    {627, 343, 773, 959, 943},
    {767, 473, 103, 699, 303}
};
```

Solution (my first attempt)

```
    int max_sum(const std::vector<std::vector<int>>& mat)
    {
        if (mat.size()==1) return mat[0][0];
        int ans = std::numeric_limits<int>::min();

        for(int y=0; y!=mat.size(); ++y)
        {
            int tmp = mat[y][mat.size()-1];
            std::vector<std::vector<int>> sub_mat;
            for(int yy=0; yy!=mat.size(); ++yy)
            {
                if (yy!=y)
                {
                    auto row = mat[yy];
                    row.pop_back();
                    sub_mat.push_back(row);
                }
            }

            int sub = max_sum(sub_mat);
            if (ans < tmp + sub) ans = tmp + sub;
        }
        return ans;
    }
```

If the matrix is *N by N*, then the complexity is *f(N)*, we have :

$$f(N) = O(N * f(N-1))$$
$$f(N) = O(N!)$$

Can we optimize it? As we increase the size to 15 x 15, the above implementation does not work. This problem is known as assignment problem, solved by Hungarian algorithm.