KM算法

```
struct KM{
#define inf 0x3f3f3f3f3f3f3f3f
    vector<vector<int>> mp;
    int n;
    KM(int cntn){
        n = cntn;
        mp.resize(n + 1);
        for(int i = 1;i <= n;i ++) mp[i].resize(n + 1, -inf);
    }
    void addedge(int u, int v, int w){
        mp[u][v] = w;
    }
    int solve(){
        vector<int> match(n + 1, -1);
        vector<int> ex(n + 1), ey(n + 1);
        for(int i = 1;i <= n;i ++){
            vector<int> visy(n + 1);
            auto Match = [&](int node) -> void {
                int x, y = 0, yy = 0, delta;
                vector<int> pre(n + 1), slack(n + 1, inf);
                match[y] = node;
                while(1){
                    x = match[y], delta = inf, visy[y] = 1;
                    for(int i = 1;i <= n;i ++){
                        if(visy[i]) continue;
                        if(slack[i] > ex[x] + ey[i] - mp[x][i]){
                            slack[i] = ex[x] + ey[i] - mp[x][i];
                            pre[i] = y;
                        }
                        if(slack[i] < delta) delta = slack[i], yy = i;
                    }
                    for(int i = 0;i <= n;i ++){
                        if(visy[i]) ex[match[i]] -= delta, ey[i] += delta;
                        else slack[i] -= delta;
                    }
                    y = yy;
                    if(match[y] == -1) break;
                }
                while(y) match[y] = match[pre[y]], y = pre[y];
            };
            Match(i);
        }
        int ans = 0;
        for(int i = 1;i <= n;i ++) if(match[i] != -1) ans += mp[match[i]][i];
        return ans;
    }
};
```

mp数组建图左边单向指向右边

$match[i]$数组代表与右边i节点相匹配的左边节点编号