图论

建图方法

分层图

- P4568飞行路线

  问题描述：求两点间最短路，其中允许不消耗时间经过 k 条边，$1 \le k \le 10$。

  将图分为 $k+1$ 层，分别表示已经跳过了 i 条边，即 u_i, v_i 有边相连，则令 u_i, v_{i+1} 连一条 0 长度的边。

  在分层图上 dijkstra 即可，$O((n+m)k\log(mk))$。

加点

人为加入超级源点，超级汇点连接所有点，以便套用网络流/最短路模型。

线段树建图优化

线段树优化建图是一种优化图中边的数量的技巧。主要处理边数量极多但存在规律（一个点向一个区间所有点连边）的问题。

这类问题的主要特点是题型思路简单但数据范围巨大，建好图以后就是一些常见问题（拓扑排序/最短路/网络流）。

建图方法大概就是把图建成线段树就好了（

建一颗“出树”，边的方向为从子节点指向父节点；再建一颗“入树”，边的方向从父节点指向子节点。

如果某个点要与区间连边，就在出树中找到这个点，向入树中的区间节点依次连边。

区间与点连边同理。

- CF786B Legacy

最短路算法的应用

floyd

- 不断向点集 S 添加点，求每次添加点后，只经过 S 中的点，某两点间的最短距离。

  例题：P1119 灾后重建 、 ABC208D

  考虑 floyd 的意义：最外层枚举 k ， $f_{u,v}$ 表示只经过编号在 $[1,k]$ 内的点，两点间的最短路。

  所以把点按输入顺序重新编号，逐次跑 flody 即可。

- 传递闭包：01矩阵 A 满足 $A_{u,v}$ 表示 u 是否能够抵达 v ，求 A 。

```
for(int k=1;k<=n;++k)
    for(int i=1;i<=n;++i)
        for(int j=1;j<=n;++j)
            g[i][j]&=g[i][k] && g[k][j];
```

例题：图函数。

dijkstra

- 多起点多终点最短路

  从某些点中选择一点出发，到达任何一个终点，求最短路。

  建立超级源点指向所有起点建立边权为 0 的边，求从源点开始的全源最短路即可。

  容易发现其实超级源点是不用建的，一开始把所有起点都扔到优先队列里就行。

- 差分约束系统

  给出关于x_i 的 n 元不等式组 $\begin{cases} x_{c_1} - x_{c'_1} \le y_1 \\ x_{c_2} - x_{c'_2} \le y_2 \\ \dots \\ x_{c_m} - x_{c'_m} \le y_m \end{cases}$ ，求其一个解。

  变形为 $x_{c_i} \le x_{c'_i} + y_i$ ，发现其与最短路的三角不等式相同：$dis_u \le dis_v + w_{u,v}$ 。

  也就是说，建立 $n+1$ 个点，其中 x_{c_i} 与 $x_{c'_i}$ 距离为 y_i ，超级源点 0 向每个点连 0 长度的边，求出最短路 dis_i ，则 $x_i = dis_i$.

  另：

  - $x_{c_1} - x_{c'_1} \ge y_1$ 可以两边乘 -1 .
  - $x_{c_1} - x_{c'_1} = y_1$ 等价于 $x_{c_1} - x_{c'_1} \ge y_1$ 且 $x_{c_1} - x_{c'_1} \le y_1$.

拓扑排序 & DAG 递推

有向无环图（DAG）上的问题，通常要求我们按一定顺序访问图上的点，使得每个点被访问时，所有能抵达它的点都被访问过，我们称这样的顺序为拓扑序。

拓扑排序

求拓扑序的过程称为拓扑排序。

我们每访问一个点，就删除它连出的所有边，则能抵达某个点的点都被访问过等价于其入度为 0 。

删边其实就是让与它相连的点度数减一，不需要真的把边删了。

```
void topsort(){
    queue<int> q;
    for(int i=1;i<=n;++i)
        if(!deg[i])q.push(i),tim[i]=++cnt;
    while(!q.empty()){
        int h=q.front();
        for(int i=head[h];i;i=nxt[i]){
            deg[to[i]]--;
            if(!deg[to[i]])
                q.push(to[i]),tim[to[i]]=++cnt;
        }
    }
}
```

- [P4017 最大食物链计数]

  求 DAG 上从起点到终点的最长路

  考虑递推，$f_u = \max\{f_v\} + 1$

  按拓扑顺序递推即可，注意求拓扑序和递推可以同时进行。

```
void topsort(){
    queue<int> q;
    for(int i=1;i<=n;++i)
        if(!deg[i])q.push(i);
    while(!q.empty()){
        int h=q.front();
        for(int i=head[h];i;i=nxt[i]){
            deg[to[i]]--;
            f[to[i]]=max(f[to[i]],f[h]+1);
            if(!deg[to[i]])
                q.push(to[i]);
        }
    }
}
```

- [[NOIP2020] 排水系统]

  各个入水口有一吨污水，每个节点会把流向它的污水均匀分到各个管道，求最终每个出水口排除的水的体积

  注意分数的处理和高精度。

- [[CSP-S2020] 函数调用]

  存在三种函数：全局乘法，单点加法，调用其他函数的函数（不递归），求按顺序执行所有函数后序列的值。

  容易发现所有函数构成 DAG。

  - 若只有单点加法，可以从源点出发，计算每一个函数的调用次数。有 $f_u = \sum\limits_{v \to u} f_v$ 。
  - 现在加法与乘法混用，每执行一次乘法，相当于之前的函数的执行次数乘 k ，设执行每个函数相当于序列乘以 g_i

    有 $g_u = \prod g_{son}$ ，对于乘法函数，设参数为 x ，则 $g_u = x$ ，按拓扑序逆推即可。

得到 g_u 后，我们再次计算调用次数，有 $f_u=\sum\limits_{v\to u}(f_v\prod\limits_{k>u,v\to k}g_k)$

二分图

图中的点可以被分为 U,V 两部分满足 U 中的点互相不连边，V 同理，则称该图为二分图。

- 判断

  将图染成黑白两色，互相连边的两点颜色不同，钦定一个点为黑色，开始 dfs 染色，若发现两个相邻点同色则不是二分图。

- 匹配

  取 k 条边使得任意两边不共用顶点，称这 k 条边为二分图的一个匹配。

  k 最大时称为二分图的最大匹配。

  使用匈牙利算法计算一个二分图的最大匹配，复杂度 $O(ne+m)$ ，$n=|U|,m=|V|$ 。

  过程：我们试图找到 u 的对应点，若某点 v 未被匹配，则令其与 v 匹配，否则令原先与 v 匹配的点 s 找到新的匹配。若 s 没有新的匹配，则令 u 与其他点匹配。

  正确性与复杂度证明均略。

  - 实现：

    ```
    int Match(){
        int ans=0;
        for(int i=1;i<=n;++i){
            memset(vis,0,sizeof(vis));
            if(dfs(i))ans++;
        }
        return ans;
    }

    bool dfs(int u){
        if(vis[u])return  false;
        vis[u]=1;
        for(int i=0;i<=e[u].size();++i)
            if((!match[e[u][v]])||dfs(match[e[u][v]])){
                match[e[u][v]]=u;
                return true;
            }
        return false;
    }
    ```

- 应用

  [P2417 课程]

给出每个学生可以在哪些教室上课，令每个学生挑一间教室上课，试判断是否有一种方案使每个教室都有学生。

教室与学生构成二分图，求最大匹配，判断是否等于教室数即可。