图论

positive1

2024 年 2 月 18 日

路径压缩变种

```
//Splitting
int find(int id)
{
    while(fa[id]!=id)
    {
        int tmp=fa[id];
        fa[id]=fa[fa[id]];
        id=tmp;
    }
    return id;
}
```

路径压缩变种

```
//Halving
int find(int id)
{
    while(fa[id]!=id) id=fa[id]=fa[fa[id]];
    return id;
}
```

路径压缩变种

```
//Halving
int find(int id)
{
    while(fa[id]!=id) id=fa[id]=fa[fa[id]];
    return id;
}
```

朴素的路径压缩需要递归。这种两种做法不用递归，而且最坏时间复杂度和路径压缩相同 (Robert E. Tarjan and Jan van Leeuwen 1984)。

调用 `find(u)` 后 `u` 的深度大约减半，注意 `fa[u]` 不保证指向根。

不用额外空间的按秩/大小合并

```
int find(int id)
{
    return fa[id]<0?id:fa[id]=find(fa[id]);
}
```

fa 对于根节点存储秩/大小的相反数，对于其他节点则照常存储父亲。

按随机权值合并

```
void merge(int u,int v)
{
    u=find(u),v=find(v);
    if(u==v) return;
    if(w[u]>w[v]) swap(u,v);
    fa[u]=v;
}
```

w 是事先生成的随机权值。在没有路径压缩的情况下，n 个节点的期望树高是 $O(\log n)$。

论文摘抄

定义 Ackermann 函数

$$A(i,j) = \begin{cases} 2^j & i = 1 \\ A(i-1,2) & i \geq 2, j = 1 \\ A(i-1, A(i,j-1)) & i,j \geq 2 \end{cases}$$

定义 $\alpha(m,n) = \min\{i \geq 1 | A(i, \lfloor m/n \rfloor) > \log n\}$。

论文摘抄

定义 Ackermann 函数

$$A(i,j)=\begin{cases}2^j & i=1\\ A(i-1,2) & i\geq 2,j=1\\ A(i-1,A(i,j-1)) & i,j\geq 2\end{cases}$$

定义 $\alpha(m,n)=\min\{i\geq 1|A(i,\lfloor m/n\rfloor)>\log n\}$。

如果 $\lfloor m/n\rfloor\geq 1+\log\log n$，则 $\alpha(m,n)\leq 1$；如果 $\lfloor m/n\rfloor\geq\log^*\log n$，则 $\alpha(m,n)\leq 2$。

论文摘抄

定义 Ackermann 函数

$$A(i,j)=\begin{cases}2^j & i=1\\ A(i-1,2) & i\geq 2,j=1\\ A(i-1,A(i,j-1)) & i,j\geq 2\end{cases}$$

定义 $\alpha(m,n)=\min\{i\geq 1|A(i,\lfloor m/n\rfloor)>\log n\}$。

如果 $\lfloor m/n\rfloor\geq 1+\log\log n$，则 $\alpha(m,n)\leq 1$；如果 $\lfloor m/n\rfloor\geq\log^*\log n$，则 $\alpha(m,n)\leq 2$。

设 n 是节点数，m 是 `find` 的调用次数（注意不是边数）。`fa[u]=v` 不影响复杂度分析，只要操作前 `u,v` 都是树根。大部分情况下可以认为 $m\geq n$。

论文摘抄

定义 Ackermann 函数

$$A(i,j)=\begin{cases}2^j & i=1\\ A(i-1,2) & i\geq 2,j=1\\ A(i-1,A(i,j-1)) & i,j\geq 2\end{cases}$$

定义 $\alpha(m,n)=\min\{i\geq 1|A(i,\lfloor m/n\rfloor)>\log n\}$。

如果 $\lfloor m/n\rfloor\geq 1+\log\log n$，则 $\alpha(m,n)\leq 1$；如果 $\lfloor m/n\rfloor\geq\log^*\log n$，则 $\alpha(m,n)\leq 2$。

设 n 是节点数，m 是 `find` 的调用次数（注意不是边数）。`fa[u]=v` 不影响复杂度分析，只要操作前 `u,v` 都是树根。大部分情况下可以认为 $m\geq n$。

只使用路径压缩的时间复杂度是 $\Theta(m\log_{1+m/n}n)$，结合了路径压缩与按秩/大小合并的时间复杂度是 $\Theta(m\alpha(m,n))$。

SOJ613 小 ω 的图

给定 n 个节点 m 条边的无向连通图，有边权，可能有重边自环，求 $1 \to n$ 路径权值按位与最大值。

$2 \le n \le 10^5$，$1 \le m \le 5 \times 10^5$，边权 $[0, 2^{63} - 1]$。

题解

从高往低按位贪心。当前尝试的 x 是答案的子集，当且仅当保留所有权值为 x 的超集的边后，节点 $1, n$ 连通。

题解

从高往低按位贪心。当前尝试的 x 是答案的子集，当且仅当保留所有权值为 x 的超集的边后，节点 $1, n$ 连通。

Extra tests 很猛，用 BFS/DFS 有可能 TLE，一些时间复杂度较劣的并查集也被卡了。

DFS 树

先考虑连通无向图。原图中的边要么是 DFS 树上的边，要么是反祖边。

建立 DFS 树同时得到 $dfn(u)$ 表示节点 u 是第几个被遍历的，$low(u)$ 表示和 u 的子树（包括 u）相邻的节点的 dfn 最小值。

边双连通分量

边双连通图是删去其中任意一条边，所有节点仍然连通的图。边双连通分量是极大的边双连通子图。

边双连通分量一定是 DFS 树上的连通块，并且两个边双连通分量不可能相交。我们希望在深度最浅的节点上统计边双连通分量，这个节点应当满足 $dfn(u) \leq low(u)$。注意，如果不考虑重边，求 $low(u)$ 的时候应该认为 u 和 u 的父节点不相邻，否则只有根节点满足 $dfn(u) \leq low(u)$。

边双连通分量

边双连通图是删去其中任意一条边，所有节点仍然连通的图。边双连通分量是极大的边双连通子图。

边双连通分量一定是 DFS 树上的连通块，并且两个边双连通分量不可能相交。我们希望在深度最浅的节点上统计边双连通分量，这个节点应当满足 $dfn(u) \leq low(u)$。注意，如果不考虑重边，求 $low(u)$ 的时候应该认为 u 和 u 的父节点不相邻，否则只有根节点满足 $dfn(u) \leq low(u)$。

确定每个边双连通分量的最浅点后，DFS 树自然被分成了若干个连通块，每个连通块都是边双连通分量。可以在 DFS 同时用栈维护所有不确定属于哪个边双连通分量的节点。遍历一个节点就先入栈。如果确定一个最浅点，就把它子树内没出栈的节点划进一个边双连通分量并出栈。

边双连通分量

边双连通图是删去其中任意一条边，所有节点仍然连通的图。边双连通分量是极大的边双连通子图。

边双连通分量一定是 DFS 树上的连通块，并且两个边双连通分量不可能相交。我们希望在深度最浅的节点上统计边双连通分量，这个节点应当满足 $dfn(u) \leq low(u)$。注意，如果不考虑重边，求 $low(u)$ 的时候应该认为 u 和 u 的父节点不相邻，否则只有根节点满足 $dfn(u) \leq low(u)$。

确定每个边双连通分量的最浅点后，DFS 树自然被分成了若干个连通块，每个连通块都是边双连通分量。可以在 DFS 同时用栈维护所有不确定属于哪个边双连通分量的节点。遍历一个节点就先入栈。如果确定一个最浅点，就把它子树内没出栈的节点划进一个边双连通分量并出栈。

两端点不属于同一个边双连通分量的边就是割边。n 个节点的图最多有几条割边？

点双连通分量与圆方树

点双连通图是删去其中任意一个节点，其余节点仍然连通的图。点双连通分量是极大的点双连通子图。

点双连通分量一定是 DFS 树上的连通块，但是两个点双连通分量可以有一个公共点。我们仍然在深度最浅的节点上统计点双连通分量，只是一个节点上可能有好几个点双连通分量。

点双连通分量与圆方树

点双连通图是删去其中任意一个节点，其余节点仍然连通的图。点双连通分量是极大的点双连通子图。

点双连通分量一定是 DFS 树上的连通块，但是两个点双连通分量可以有一个公共点。我们仍然在深度最浅的节点上统计点双连通分量，只是一个节点上可能有好几个点双连通分量。

圆方树用圆点表示原图上的点，方点表示点双连通分量。圆点和方点有边，当且仅当方点对应的点双连通分量包含原图上圆点的对应点。因为 DFS 树是树，所以圆方树是树。

点双连通分量与圆方树

点双连通图是删去其中任意一个节点，其余节点仍然连通的图。点双连通分量是极大的点双连通子图。

点双连通分量一定是 DFS 树上的连通块，但是两个点双连通分量可以有一个公共点。我们仍然在深度最浅的节点上统计点双连通分量，只是一个节点上可能有好几个点双连通分量。

圆方树用圆点表示原图上的点，方点表示点双连通分量。圆点和方点有边，当且仅当方点对应的点双连通分量包含原图上圆点的对应点。因为 DFS 树是树，所以圆方树是树。

对于 u 和它的儿子 v，如果 $low(v) \geq dfn(u)$，则 u, v 确定了一个点双连通分量（因为删去 u 使得 v 和 v 子树外的所有点不连通）。新建方点 w，w 在圆方树上的父亲是 u，并把 v 子树内没确定圆方树上的父亲的节点的父亲设为 w。这个过程也可以用栈维护。

点双连通分量与圆方树

圆方树上度数大于 1 的圆点就是割点。

无向图上的节点 u, v 间所有简单路径的交集是 u, v 在圆方树上路径经过的圆点，所有简单路径的并集是 u, v 在圆方树上路径经过的方点的邻居。（为什么？）

点双连通分量与圆方树

圆方树上度数大于 1 的圆点就是割点。

无向图上的节点 u, v 间所有简单路径的交集是 u, v 在圆方树上路径经过的圆点，所有简单路径的并集是 u, v 在圆方树上路径经过的方点的邻居。（为什么？）

如何找到包含原图中一条边两端点的点双连通分量？

CF1916F Group Division

集合 S 的导出子图是以 S 为点集，以原图上两端点均在 S 中的边为边集的图。给定 $n = n_1 + n_2$ 个节点 m 条边的点双连通图，没有重边自环。求将其划分成集合 S_1, S_2 的方案，满足 $|S_1| = n_1$，$|S_2| = n_2$，且 S_1, S_2 的导出子图均连通。保证有解。T 组数据。

$1 \le T \le 1000$，$n_1, n_2, m \ge 1$，$\sum(n_1 + n_2) \le 2000$，$\sum m \le 5000$。

事实上，存在节点的排列 $\{a_n\}$，使得对任意 $1 \le i < n$，有 $\{a_1, a_2, \ldots, a_i\}, \{a_{i+1}, a_{i+2}, \ldots, a_n\}$ 的导出子图均连通。

题解

从节点 1 开始跑最朴素的 Tarjan 算法，求出 dfn, low 和 DFS 树上的父亲 fa。设 $low(u)$（DFS 序编号）对应的节点为 $up(u)$。

由于原图是点双连通图，$u \neq up(u)$（$u \neq 1$）。$up(u) = fa(u)$ 当且仅当 $fa(u) = 1$，且这样的 u 只有一个。建一张新图，每个节点 u 连边 $fa(u) \to u$，$up(u) \to u$。最后从节点 1 开始 DFS 遍历新图，有多条出边先走 dfn 较大的，按 DFS 遍历的顺序排列节点就是答案。

题解

从节点 1 开始跑最朴素的 Tarjan 算法，求出 dfn, low 和 DFS 树上的父亲 fa。设 $low(u)$（DFS 序编号）对应的节点为 $up(u)$。

由于原图是点双连通图，$u \neq up(u)$（$u \neq 1$）。$up(u) = fa(u)$ 当且仅当 $fa(u) = 1$，且这样的 u 只有一个。建一张新图，每个节点 u 连边 $fa(u) \to u$，$up(u) \to u$。最后从节点 1 开始 DFS 遍历新图，有多条出边先走 dfn 较大的，按 DFS 遍历的顺序排列节点就是答案。

先说明前缀的导出子图连通，也就是除节点 1 外每个节点都和至少一个更早遍历的节点相邻。

- 如果 u 被 $fa(u)$ 遍历，则 u 和更早遍历的 $fa(u)$ 相邻。
- 否则，$up(u)$ 是更早遍历的节点，但 u 不一定和 $up(u)$ 相邻。如果 u 不和 $up(u)$ 相邻，则 $up(u)$ 来自 u 的子树。设 u 的儿子 v 满足 $up(u) = up(v)$，由于 u, v 都是新图上 $up(u)$ 的出边且 $dfn(v) > dfn(u)$，故 v 比 u 更早遍历。

题解

再说明后缀的导出子图连通，也就是除最后一个被遍历的节点外每个节点都和至少一个更晚遍历的节点相邻。最后一个被遍历的节点是满足 $up(u) = fa(u) = 1$ 的唯一节点 u，因为 $dfn(u) = 2$，是节点 1 最后遍历的节点。其他节点均满足 $fa(u) \neq up(u)$。

- 如果 $fa(u)$ 比 $up(u)$ 早遍历，那么 u 被 $fa(u)$ 遍历，且比 $up(u)$ 早遍历。如果 u 不和 $up(u)$ 相邻，找到上文所述 u 的儿子 v，$fa(v) = u$，$up(v)$ 比 u 晚遍历，所以 v 比 u 晚遍历。
- 否则，u 被 $up(u)$ 遍历。由于 $dfn(fa(u)) < dfn(u)$，$fa(u)$ 比 u 晚遍历。

强连通分量

接下来讨论有向图。强连通图是任意一对节点互相可达的图。强连通分量是极大的强连通子图。

仍然建出 DFS 树。注意有向图不一定存在可以到达所有节点的节点，需要从不同起点开始 DFS，同时避免重复遍历。有向图上的边可能既不是树边，也不是反祖边，这样的边不影响强连通性（反证法），但可能干扰 low 的作用——$low(u)$ 对应的节点应当和 u 处于同一个强连通分量。

强连通分量

接下来讨论有向图。强连通图是任意一对节点互相可达的图。强连通分量是极大的强连通子图。

仍然建出 DFS 树。注意有向图不一定存在可以到达所有节点的节点，需要从不同起点开始 DFS，同时避免重复遍历。有向图上的边可能既不是树边，也不是反祖边，这样的边不影响强连通性（反证法），但可能干扰 low 的作用——$low(u)$ 对应的节点应当和 u 处于同一个强连通分量。

强连通分量一定是 DFS 树上的连通块，并且两个强连通分量不可能相交。我们希望在深度最浅的节点上统计强连通分量，这个节点应当满足 $dfn(u) \leq low(u)$。划分连通块的过程和边双连通分量一样。

强连通分量

接下来讨论有向图。强连通图是任意一对节点互相可达的图。强连通分量是极大的强连通子图。

仍然建出 DFS 树。注意有向图不一定存在可以到达所有节点的节点，需要从不同起点开始 DFS，同时避免重复遍历。有向图上的边可能既不是树边，也不是反祖边，这样的边不影响强连通性（反证法），但可能干扰 low 的作用——$low(u)$ 对应的节点应当和 u 处于同一个强连通分量。

强连通分量一定是 DFS 树上的连通块，并且两个强连通分量不可能相交。我们希望在深度最浅的节点上统计强连通分量，这个节点应当满足 $dfn(u) \leq low(u)$。划分连通块的过程和边双连通分量一样。

如果把强连通分量缩点，保留原图中连接不同强连通分量的边，得到有向无环图。记得去重边。

2-SAT

给出若干布尔变量和布尔变量之间的限制：$a = u \vee b = v$，其中 a, b 是布尔变量，u, v 是给定的 `true` 或 `false`。问是否存在满足所有限制的变量取值，如果存在，给出构造。

2-SAT

给出若干布尔变量和布尔变量之间的限制：$a = u \vee b = v$，其中 a, b 是布尔变量，u, v 是给定的 `true` 或 `false`。问是否存在满足所有限制的变量取值，如果存在，给出构造。

把每个变量拆成两个节点 a_0, a_1，表示两种可能的取值。以 $a = \texttt{true} \vee b = \texttt{false}$ 为例，连边 $a_0 \to b_0, b_1 \to a_1$。

如果某个变量 a 满足 a_0, a_1 属于同一个强连通分量，就是无解。取 a 为 a_0, a_1 在缩点后的图上拓扑序靠后者，（根据边的对称性）能构造出一种合法方案。DFS 过程越早求出的强连通分量，在缩点后的图上拓扑序越靠后。

2-SAT

给出若干布尔变量和布尔变量之间的限制：$a = u \vee b = v$，其中 a, b 是布尔变量，u, v 是给定的 `true` 或 `false`。问是否存在满足所有限制的变量取值，如果存在，给出构造。

把每个变量拆成两个节点 a_0, a_1，表示两种可能的取值。以 $a = \texttt{true} \vee b = \texttt{false}$ 为例，连边 $a_0 \to b_0, b_1 \to a_1$。

如果某个变量 a 满足 a_0, a_1 属于同一个强连通分量，就是无解。取 a 为 a_0, a_1 在缩点后的图上拓扑序靠后者，（根据边的对称性）能构造出一种合法方案。DFS 过程越早求出的强连通分量，在缩点后的图上拓扑序越靠后。

每条限制都是只涉及两个变量的布尔表达式，可以通过分类讨论真值表解决。如果要求 $a = \texttt{true}$，就连边 $a_0 \to a_1$。

二分图最大匹配

二分图是节点可以划分为两个集合（左部点，右部点），且两个集合内部没有边的图。匹配就是选出一些边，使得这些边没有公共顶点。

二分图两组匹配的异或只可能由环和链组成。任意合法匹配都可以通过不断反转环与链的匹配状态得到。交错路是可以将匹配状态取反的链，交错环是可以将匹配状态取反的环。

增广路算法

从左部未匹配点出发 DFS，从左向右走非匹配边，从右向左走匹配边。如果从右向左没有匹配边，就是走到了右部未匹配点，找到了增广路。把左部未匹配点到右部未匹配点路径上的边匹配状态取反，使得匹配数量增加 1。

这个算法不会把匹配点转化成未匹配点，所以每个左部点各尝试一次增广后就找到了最大匹配。对于 n 个节点，m 条边的二分图，时间复杂度为 $O(n(n+m))$。

最小点覆盖与最大独立集

覆盖集是节点的集合，其包含每条边两端的至少一个节点。

容易发现最小点覆盖 $\geq$ 最大匹配。并且可以构造取等的方案。

最小点覆盖与最大独立集

覆盖集是节点的集合，其包含每条边两端的至少一个节点。

容易发现最小点覆盖 $\geq$ 最大匹配。并且可以构造取等的方案。

先求出最大匹配。再从左部所有未匹配点出发 DFS，从左向右走非匹配边，从右向左走匹配边。此时不可能走到右部未匹配点。没遍历到的左部点和遍历到的右部点是最小点覆盖的一种方案。（为什么？）

最小点覆盖与最大独立集

覆盖集是节点的集合，其包含每条边两端的至少一个节点。

容易发现最小点覆盖 $\geq$ 最大匹配。并且可以构造取等的方案。

先求出最大匹配。再从左部所有未匹配点出发 DFS，从左向右走非匹配边，从右向左走匹配边。此时不可能走到右部未匹配点。没遍历到的左部点和遍历到的右部点是最小点覆盖的一种方案。（为什么？）

独立集是节点的集合，其包含每条边两端的至多一个节点。独立集就是覆盖集的补集。最大独立集就是总点数减最小点覆盖。

必经点与可行点

如果找到了一条从左部未匹配点 u 到左部匹配点 v 的交错路，就可以把路径上的边匹配状态取反，于是 u 是可行点，v 不是必经点。

必经点与可行点

如果找到了一条从左部未匹配点 u 到左部匹配点 v 的交错路，就可以把路径上的边匹配状态取反，于是 u 是可行点，v 不是必经点。

从左部所有未匹配点出发 DFS，从左向右走非匹配边，从右向左走匹配边，可以遍历所有不是左部必经点的 v。

从左部所有匹配点出发 DFS，从左向右走匹配边，从右向左走非匹配边，可以遍历所有左部可行点 u。

必经点与可行点

如果找到了一条从左部未匹配点 u 到左部匹配点 v 的交错路，就可以把路径上的边匹配状态取反，于是 u 是可行点，v 不是必经点。

从左部所有未匹配点出发 DFS，从左向右走非匹配边，从右向左走匹配边，可以遍历所有不是左部必经点的 v。

从左部所有匹配点出发 DFS，从左向右走匹配边，从右向左走非匹配边，可以遍历所有左部可行点 u。

右部点同理。

必经边与可行边

如果找到了一个交错环，就可以把环上的边匹配状态取反，于是环上所有边都不是必经边，且都是可行边。

必经边与可行边

如果找到了一个交错环，就可以把环上的边匹配状态取反，于是环上所有边都不是必经边，且都是可行边。

将图定向为从左向右走非匹配边，从右向左走匹配边，此时图上的所有环一定都是交错环。一条边属于某个交错环当且仅当两端点属于同一个强连通分量，把这样的边标为非必经边和可行边。

当然可行边还要包括初始匹配边。

二分图博弈

给定二分图和棋子的起点。双方交替进行游戏，每次将棋子沿着边移动一步，不能经过重复的点。不能移动者输。求先手是否有必胜策略。

二分图博弈

给定二分图和棋子的起点。双方交替进行游戏，每次将棋子沿着边移动一步，不能经过重复的点。不能移动者输。求先手是否有必胜策略。

把最大匹配作为行动纲领，如果一方首先将棋子到了某条匹配边的一端上，另一方就可以将棋子移到匹配边的另一端。最后肯定走出一条交错路，胜负已判。

二分图博弈

给定二分图和棋子的起点。双方交替进行游戏，每次将棋子沿着边移动一步，不能经过重复的点。不能移动者输。求先手是否有必胜策略。

把最大匹配作为行动纲领，如果一方首先将棋子到了某条匹配边的一端上，另一方就可以将棋子移到匹配边的另一端。最后肯定走出一条交错路，胜负已判。

一个起点是先手必胜的当且仅当这个点是最大匹配必经点。

必要性：若存在一个最大匹配不经过起点，那么先手每次操作时，要么无路可走，要么走到了某个匹配点上（否则就找到了一条增广路）。于是，后手只要每次走匹配边即可。

充分性：从起点 u 沿着匹配边移动到 v，v 一定不是删去 u 的二分图的最大匹配必经点（因为删去 u 的二分图和删去 u, v 的二分图最大匹配大小相同），由必要性，先手必胜。

P3731 [HAOI2017] 新型城市化

团是一个点集，满足其中任意两个节点都相邻。给定 n 个节点的无向完全图，从中挖掉 m 条边。保证此时图可以划分为最多两个团。问 m 条边中哪些边满足加入这条边后图的最大团大小比原来大。

$1 \leq n \leq 10^4$，$0 \leq m \leq \min\{1.5 \times 10^5, \frac{n(n-1)}{2}\}$。

题解

考虑补图。原图可以划分为最多两个团，补图就是二分图。原图最大团就是补图最大独立集。二分图最大独立集是总点数减最大匹配。最大团增大说明最大匹配减小。所以就是问 m 条边中哪些是最大匹配必经边。

Hall 定理

记二分图左部点集为 V_1，右部点集为 V_2，$N(S)$ 表示和集合 S 中的某些节点相邻的节点集合（$S=\{u\}$ 就简写作 $N(u)$），则存在大小为 $|V_1|$ 的匹配（完美匹配）当且仅当 $\forall S \subseteq V_1, |S| \leq |N(S)|$。

Hall 定理

记二分图左部点集为 V_1，右部点集为 V_2，$N(S)$ 表示和集合 S 中的某些节点相邻的节点集合（$S=\{u\}$ 就简写作 $N(u)$），则存在大小为 $|V_1|$ 的匹配（完美匹配）当且仅当 $\forall S \subseteq V_1, |S| \leq |N(S)|$。

只证充分性。$|V_1|=1$ 显然成立。假设 $|V_1|<n$ 成立，当 $|V_1|=n$ 时，

- 如果 $\exists S \subsetneq V_1, S \neq \varnothing, |S|=|N(S)|$，子图 $(S, N(S))$ 和 $(V_1-S, V_2-N(S))$ 均满足归纳假设。
- 否则，任取 $u \in V_1, v \in N(u)$，子图 $(V_1-\{u\}, V_2-\{v\})$ 满足归纳假设。

Hall 定理

记二分图左部点集为 V_1，右部点集为 V_2，$N(S)$ 表示和集合 S 中的某些节点相邻的节点集合（$S=\{u\}$ 就简写作 $N(u)$），则存在大小为 $|V_1|$ 的匹配（完美匹配）当且仅当 $\forall S\subseteq V_1, |S| \leq |N(S)|$。

只证充分性。$|V_1|=1$ 显然成立。假设 $|V_1|<n$ 成立，当 $|V_1|=n$ 时，

- 如果 $\exists S\subsetneq V_1, S\neq\varnothing, |S|=|N(S)|$，子图 $(S, N(S))$ 和 $(V_1-S, V_2-N(S))$ 均满足归纳假设。
- 否则，任取 $u\in V_1, v\in N(u)$，子图 $(V_1-\{u\}, V_2-\{v\})$ 满足归纳假设。

推广：存在大小为 $|V_1|-k$ 的匹配，当且仅当 $\forall S\subseteq V_1, |S|-k\leq|N(S)|$。

证明：V_2 新建 k 个节点，和所有 V_1 中的节点连边。此时存在大小为 $|V_1|$ 的匹配。删去新建节点的匹配即可。

P4518 [JSOI2018] 绝地反击

给定 n 个平面上的点 (x_i, y_i)，在所有将点移动到以原点为圆心、R 为半径的圆上，且恰好 n 等分圆弧的方案中，求这 n 个点移动距离最大值的最小值。

$3 \leq n \leq 200$，$0 \leq |x_i|, |y_i|, R \leq 100$。

题解

二分答案。每个点能移动到一段圆弧。钦定一个 n 等分点和圆弧的端点重合（这包含了所有本质不同的情况）。每个点向能移动到的 n 等分点连边。存在完美匹配则答案合法。

题解

二分答案。每个点能移动到一段圆弧。钦定一个 n 等分点和圆弧的端点重合（这包含了所有本质不同的情况）。每个点向能移动到的 n 等分点连边。存在完美匹配则答案合法。

区间连边的匹配问题（任意左部点 u 均满足 $N(u)$ 是连续段）可以使用 Hall 定理判定是否存在完美匹配。如果存在点集 S 满足 $|S| > |N(S)|$，则存在一个连续段 $N(S)$ 满足 $|S| > |N(S)|$，只需要检查 $O(n^2)$ 个连续段就能判定是否存在完美匹配。这种做法容易拓展到环上。

Kruskal 重构树

一开始所有重构树节点都是孤立点。在 Kruskal 算法执行过程中，如果要把权值为 w 的边 (u, v) 加入最小生成树，则新建重构树节点 x，点权为 w，（用并查集）找到 u, v 在重构树上的根节点 fu, fv，并将 fu, fv 的父亲设为 x。

如果原图有 n 个节点并且连通，最后会得到一棵 $2n-1$ 个节点的二叉树，所有原图上的节点都是 Kruskal 重构树的叶节点。从叶节点不断跳父亲，点权递增。

Kruskal 重构树

一开始所有重构树节点都是孤立点。在 Kruskal 算法执行过程中，如果要把权值为 w 的边 (u, v) 加入最小生成树，则新建重构树节点 x，点权为 w，（用并查集）找到 u, v 在重构树上的根节点 fu, fv，并将 fu, fv 的父亲设为 x。

如果原图有 n 个节点并且连通，最后会得到一棵 $2n-1$ 个节点的二叉树，所有原图上的节点都是 Kruskal 重构树的叶节点。从叶节点不断跳父亲，点权递增。

一条路径的瓶颈是其经过的边的权值最大值。节点 u, v 的最小瓶颈路（所有简单路径中瓶颈的最小值）是 u, v 在 Kruskal 重构树上 LCA 的权值。

设 u 是原图上的节点，v 是 u 的祖先，如果边权互不相同，则所有到 u 的最小瓶颈路不超过 v 的权值的节点恰好是 v 子树的所有叶节点。

Kruskal 重构树

一开始所有重构树节点都是孤立点。在 Kruskal 算法执行过程中，如果要把权值为 w 的边 (u, v) 加入最小生成树，则新建重构树节点 x，点权为 w，（用并查集）找到 u, v 在重构树上的根节点 fu, fv，并将 fu, fv 的父亲设为 x。

如果原图有 n 个节点并且连通，最后会得到一棵 $2n-1$ 个节点的二叉树，所有原图上的节点都是 Kruskal 重构树的叶节点。从叶节点不断跳父亲，点权递增。

一条路径的瓶颈是其经过的边的权值最大值。节点 u, v 的最小瓶颈路（所有简单路径中瓶颈的最小值）是 u, v 在 Kruskal 重构树上 LCA 的权值。

设 u 是原图上的节点，v 是 u 的祖先，如果边权互不相同，则所有到 u 的最小瓶颈路不超过 v 的权值的节点恰好是 v 子树的所有叶节点。

怎么求最大瓶颈路？

SOJ64 【STR #2】雪之国度

给定 n 个节点 m 条边的无向连通图，有点权 W，可能有重边自环。边 (u, v) 的权值是 $|W_u - W_v|$。q 次询问，每次给定节点 u, v，问两条从 u 到 v 的边不交简单路径的瓶颈的最大值的最小值，如果不存在，输出 `infinitely`。

$1 \leq n, q \leq 10^5$，$1 \leq m \leq 5 \times 10^5$，$0 \leq W_i \leq 2 \times 10^5$。

题解

用 Kruskal 算法的思路求“最小生成边双”。如果 $u = v$ 或 u, v 不属于同一个边双连通分量，输出 `infinitely`。

先建出最小生成树，其边权不影响答案。然后为最小生成树上的每条边找到权值最小的非树边替代方案，以此作为树的边权。节点 u, v 树上简单路径的最大边权就是答案。

按权值从小到大枚举非树边，树上每条边的权值只更新一次。在一条边权值更新后把较深的端点合并到较浅的端点。更新权值的过程就是不断暴力跳父亲，更新第一个没被合并的点。

题解

用 Kruskal 算法的思路求“最小生成边双”。如果 $u = v$ 或 u, v 不属于同一个边双连通分量，输出 `infinitely`。

先建出最小生成树，其边权不影响答案。然后为最小生成树上的每条边找到权值最小的非树边替代方案，以此作为树的边权。节点 u, v 树上简单路径的最大边权就是答案。

按权值从小到大枚举非树边，树上每条边的权值只更新一次。在一条边权值更新后把较深的端点合并到较浅的端点。更新权值的过程就是不断暴力跳父亲，更新第一个没被合并的点。

树链最大值有一种好写的启发式合并做法。按权值从小到大枚举树边，把较小的连通块的父亲设为较大的连通块，只要经过这条边答案就和树边权值取 max。这样就建出了树高 $O(\log n)$ 的“重构树”，满足任意两点间的树链最大值和原树相同。

Boruvka

Boruvka 算法可以求最小生成树。初始所有节点都是孤立的连通块。一轮迭代中，先为每个连通块找到权值最小的出边，再枚举所有找到的边，如果两端还未连通，就合并到一个连通块并加入最小生成树。每一轮迭代连通块个数减小至少一半。n 个节点的图经过 $O(\log n)$ 轮迭代算法就会结束。

Boruvka

Boruvka 算法可以求最小生成树。初始所有节点都是孤立的连通块。一轮迭代中，先为每个连通块找到权值最小的出边，再枚举所有找到的边，如果两端还未连通，就合并到一个连通块并加入最小生成树。每一轮迭代连通块个数减小至少一半。n 个节点的图经过 $O(\log n)$ 轮迭代算法就会结束。

该算法可以处理边数不可直接枚举的图。找权值最小出边与合并操作分离，使得维护边的数据结构只需要面对 $O(\log n)$ 轮修改，相比之下，专注于一个连通块的 Prim 算法就是强制在线的 $O(n)$ 次修改。

期望线性最小生成树

设图有 n 个节点 m 条边。

先进行两轮 Boruvka 算法中的迭代，使节点数降至 $\frac{n}{4}$。

在缩点后的图上以 $\frac{1}{2}$ 的概率随机选边，递归跑最小生成树（森林）。如果一条边在边子集的最小生成树上不优，则在边全集的最小生成树上不优。删去所有不优的边后期望边数不超过 $\frac{n}{2}$。

最后对剩下的边递归跑最小生成树。

期望时间复杂度 $T(n,m) = T(\frac{n}{4}, \frac{m}{2}) + T(\frac{n}{4}, \frac{n}{2}) + O(n+m) = O(n+m)$。

P9701 [GDCPC2023] Classic Problem

给定 n 个节点的无向完全图与 m 个三元组 (u_i, v_i, w_i)。节点 x, y（$1 \le x < y \le n$）之间的边权如下：

- 如果存在 i 满足 $u_i = x, v_i = y$，则边权为 w_i。
- 否则边权为 $y - x$。

求最小生成树的边权和。T 组数据。

$1 \le n \le 10^9$，$0 \le m \le 10^5$，$1 \le T \le 10^5$，$\sum m \le 5 \times 10^5$，$0 \le w_i \le 10^9$，$1 \le u_i < v_i \le n$，$\forall i \ne j, (u_i, v_i) \ne (u_j, v_j)$。

题解

m 条特殊边最多覆盖 $2m$ 个节点，连续的没连特殊边的点显然用权值为 1 的边串起来。这样点数是 $O(m)$。

题解

m 条特殊边最多覆盖 $2m$ 个节点，连续的没连特殊边的点显然用权值为 1 的边串起来。这样点数是 $O(m)$。

然后使用 Boruvka 算法。先用 m 条特殊边更新最小出边。再对每个节点 i 求出前后第一个不属于同一连通块且没和 i 连特殊边的节点，用 $y-x$ 更新最小出边。

题解

m 条特殊边最多覆盖 $2m$ 个节点，连续的没连特殊边的点显然用权值为 1 的边串起来。这样点数是 $O(m)$。

然后使用 Boruvka 算法。先用 m 条特殊边更新最小出边。再对每个节点 i 求出前后第一个不属于同一连通块且没和 i 连特殊边的节点，用 $y-x$ 更新最小出边。

每个节点都处理出前一个不属于同一连通块的节点。如果不幸和 i 连了特殊边，就暴力往下一个节点跳。如果下一个节点又和 i 属于同一连通块，就再跳到“前一个不属于同一连通块的节点”，如此循环。跳的总次数是 $O(m)$ 的。

三元环

给定 n 个节点 m 条边的无向图，没有重边自环，求所有三元环。

三元环

给定 n 个节点 m 条边的无向图，没有重边自环，求所有三元环。

将节点按照度数从小到大排序，度数相同的顺序任意。记节点 u 排序后的所在下标为 $rank(u)$。

枚举三元环中 $rank$ 最大者 u，标记所有和 u 相邻的节点，再枚举和 u 相邻的节点 v（$rank(u) > rank(v)$），枚举和 v 相邻的节点 w（$rank(v) > rank(w)$），如果 w 被 u 标记过，就找到了三元环。

三元环

考虑 v 贡献的时间复杂度。设 v 的度数为 $d(v)$。

- 如果 $d(v) \le \sqrt{m}$，则 (u, w) 总数为 $O(d^2(v))$，又因为 $\sum d(i) = 2m$，所以 $\sum_{d(i)\le\sqrt{m}} d^2(i) = O(m\sqrt{m})$。
- 否则，因为 $d(u) \ge d(v) > \sqrt{m}$，u 只能有 $O(\sqrt{m})$ 个，$\sum_{d(i)>\sqrt{m}} d(i)\sqrt{m} = O(m\sqrt{m})$。

三元环

考虑 v 贡献的时间复杂度。设 v 的度数为 $d(v)$。

- 如果 $d(v) \le \sqrt{m}$，则 (u, w) 总数为 $O(d^2(v))$，又因为 $\sum d(i) = 2m$，所以 $\sum_{d(i)\le\sqrt{m}} d^2(i) = O(m\sqrt{m})$。
- 否则，因为 $d(u) \ge d(v) > \sqrt{m}$，u 只能有 $O(\sqrt{m})$ 个，$\sum_{d(i)>\sqrt{m}} d(i)\sqrt{m} = O(m\sqrt{m})$。

这同时说明了三元环个数是 $O(m\sqrt{m})$（完全图就能卡满），可以支持比计数更复杂的操作。

三元环

考虑 v 贡献的时间复杂度。设 v 的度数为 $d(v)$。

- 如果 $d(v) \le \sqrt{m}$，则 (u, w) 总数为 $O(d^2(v))$，又因为 $\sum d(i) = 2m$，所以 $\sum_{d(i)\le\sqrt{m}} d^2(i) = O(m\sqrt{m})$。
- 否则，因为 $d(u) \ge d(v) > \sqrt{m}$，u 只能有 $O(\sqrt{m})$ 个，$\sum_{d(i)>\sqrt{m}} d(i)\sqrt{m} = O(m\sqrt{m})$。

这同时说明了三元环个数是 $O(m\sqrt{m})$（完全图就能卡满），可以支持比计数更复杂的操作。

要求 $rank(v) > rank(w)$ 是为了避免算重，只要 $rank(u) > rank(v)$ 就能保证时间复杂度正确。

四元环计数

枚举四元环中 $rank$ 最大者 u。注意四元环中 $rank$ 最小者可能和 u 相邻也可能和 u 相对。如果和 u 相邻的节点 v, w 有共同相邻的节点 x（$rank(u) > \max\{rank(v), rank(w), rank(x)\}$），就形成四元环。

四元环计数

枚举四元环中 $rank$ 最大者 u。注意四元环中 $rank$ 最小者可能和 u 相邻也可能和 u 相对。如果和 u 相邻的节点 v, w 有共同相邻的节点 x（$rank(u) > \max\{rank(v), rank(w), rank(x)\}$），就形成四元环。

枚举和 u 相邻的节点 v（$rank(u) > rank(v)$），枚举和 v 相邻的节点 x（$rank(u) > rank(x)$），所有之前访问到 x 的方案都可以和 $u - v - x$ 拼成四元环，将 x 的访问次数加入答案，并将 x 的访问次数增加 1。

时间复杂度也是 $O(m\sqrt{m})$。

P3547 [POI2013] CEN-Price List

给定 n 个节点 m 条边的无向连通图，边权均为 a。将原来图中满足最短路等于 $2a$ 的所有点对之间加一条权值为 b 的边。求单源最短路。

$2 \leq n \leq 10^5$，$1 \leq m \leq 10^5$，$1 \leq a, b \leq 1000$。

题解

BFS 一遍求出单位边权的最短路。如果 $b \geq 2a$，直接输出答案。

否则把最短路的相邻两个 a 换成 b 总是优的，并且合法。但这样可能有一个 a 换不掉，还要求出只走权值为 b 的边的单源最短路。

题解

BFS 一遍求出单位边权的最短路。如果 $b \geq 2a$，直接输出答案。

否则把最短路的相邻两个 a 换成 b 总是优的，并且合法。但这样可能有一个 a 换不掉，还要求出只走权值为 b 的边的单源最短路。

BFS 到节点 u 时，枚举和 u 相邻的节点 v，枚举和 v 相邻的节点 w。如果 u, w 不相邻就更新 w 的最短路。注意 BFS 最短路只需要更新一次，为保证复杂度，只要 u, w 不相邻，下次就不枚举 $?-v-w$。如果没有 u, w 相邻的情况，时间复杂度是均摊 $O(m)$。而 u, w 相邻就是找到了三元环，三元环有 $O(m\sqrt{m})$ 个。时间复杂度瓶颈是跳过所有三元环。

差分约束

给出 n 个变量和 m 条变量之间的限制：$a - b \leq w$，其中 a, b 是变量，w 是给定常数。问是否存在满足所有限制的变量取值，如果存在，给出构造。

差分约束

给出 n 个变量和 m 条变量之间的限制：$a-b\le w$，其中 a,b 是变量，w 是给定常数。问是否存在满足所有限制的变量取值，如果存在，给出构造。

把变量作为节点。$a-b\le w$，就连一条从节点 b 到节点 a 的边权为 w 的有向边。从任意一个能到达 b 的节点跑单源最短路，如果有负环就无解（把形成负环的式子左右分别相加，得到 $0<0$），否则节点 a,b 的距离满足 $a-b\le w$。

差分约束

给出 n 个变量和 m 条变量之间的限制：$a-b\le w$，其中 a,b 是变量，w 是给定常数。问是否存在满足所有限制的变量取值，如果存在，给出构造。

把变量作为节点。$a-b\le w$，就连一条从节点 b 到节点 a 的边权为 w 的有向边。从任意一个能到达 b 的节点跑单源最短路，如果有负环就无解（把形成负环的式子左右分别相加，得到 $0<0$），否则节点 a,b 的距离满足 $a-b\le w$。

没有负环的图，最短路的边数最多为 $n-1$。Bellman-Ford 算法松弛 $n-1$ 轮后如果还能松弛就是有负环。SPFA 算法可以记录一个节点当前最短路经过的边数，达到 n 就是有负环；写了 inqueue 优化的前提下，也可以记录一个节点的进队次数，因为每次进队最短路经过的边数严格递增。后者常数较大。注意，没有负环也可以把一个节点的被松弛次数卡到 $O(n^2)$。

差分约束

给出 n 个变量和 m 条变量之间的限制：$a-b\le w$，其中 a,b 是变量，w 是给定常数。问是否存在满足所有限制的变量取值，如果存在，给出构造。

把变量作为节点。$a-b\le w$，就连一条从节点 b 到节点 a 的边权为 w 的有向边。从任意一个能到达 b 的节点跑单源最短路，如果有负环就无解（把形成负环的式子左右分别相加，得到 $0<0$），否则节点 a,b 的距离满足 $a-b\le w$。

没有负环的图，最短路的边数最多为 $n-1$。Bellman-Ford 算法松弛 $n-1$ 轮后如果还能松弛就是有负环。SPFA 算法可以记录一个节点当前最短路经过的边数，达到 n 就是有负环；写了 inqueue 优化的前提下，也可以记录一个节点的进队次数，因为每次进队最短路经过的边数严格递增。后者常数较大。注意，没有负环也可以把一个节点的被松弛次数卡到 $O(n^2)$。

如果原图是强连通图，可以任选起点跑单源最短路，否则可以把 n 个节点都视为起点做松弛操作。

CF241E Flights

给定 n 个节点 m 条边的有向无环图，没有重边，保证节点 1 可达节点 n。问是否存在一种将边赋权值 1 或 2 的方案，使节点 1 到节点 n 的所有路径长度相同。如果存在，给出构造。

$2 \leq n \leq 1000$，$1 \leq m \leq 5000$。

题解

如果节点 1 到节点 n 的所有路径长度相同，那么所有位于节点 1 到节点 n 路径上的节点具有和 n 相同的性质。设 $dis(u)$ 表示节点 1 到 u 的路径长度。那么一条边 $u \to v$ 产生的限制就是 $1 \leq dis(v) - dis(u) \leq 2$。拆成 $dis(v) - dis(u) \leq 2$ 和 $dis(u) - dis(v) \leq -1$ 跑差分约束。

题解

如果节点 1 到节点 n 的所有路径长度相同，那么所有位于节点 1 到节点 n 路径上的节点具有和 n 相同的性质。设 $dis(u)$ 表示节点 1 到 u 的路径长度。那么一条边 $u \to v$ 产生的限制就是 $1 \leq dis(v) - dis(u) \leq 2$。拆成 $dis(v) - dis(u) \leq 2$ 和 $dis(u) - dis(v) \leq -1$ 跑差分约束。

注意只有位于节点 1 到节点 n 路径上的节点受到路径长度相同的限制，差分约束应当忽略其他节点。

P7515 [省选联考 2021 A 卷] 矩阵游戏

给定 $(n-1)\times(m-1)$ 的矩阵 $b_{i,j}$，问是否存在 $n\times m$ 的矩阵 $a_{i,j}$ 满足 $b_{i,j}=a_{i,j}+a_{i,j+1}+a_{i+1,j}+a_{i+1,j+1}$，且 $0\le a_{i,j}\le 10^6$。如果存在，给出构造。T 组数据。

$1\le T\le 10$，$2\le n,m\le 300$，$0\le b_{i,j}\le 4\times 10^6$。

题解

先不考虑 $0 \le a_{i,j} \le 10^6$，构造出任意一个满足 $b_{i,j} = a_{i,j} + a_{i,j+1} + a_{i+1,j} + a_{i+1,j+1}$ 的矩阵 $a_{i,j}$。

然后在满足等式的基础上调整。可以选定一行 i 和变量 x_i，让这一行的元素交替加、减 x_i，也就是 $a_{i,j} \leftarrow a_{i,j} + (-1)^j x_i$。类似地对列调整，选定一列 j 和变量 y_j，令 $a_{i,j} \leftarrow a_{i,j} + (-1)^i y_j$。只要确定 a 的第一行第一列就可以推出整个矩阵，所以这 $n+m$ 个变量已经足够调整出任何满足等式的矩阵。

题解

先不考虑 $0 \le a_{i,j} \le 10^6$，构造出任意一个满足 $b_{i,j} = a_{i,j} + a_{i,j+1} + a_{i+1,j} + a_{i+1,j+1}$ 的矩阵 $a_{i,j}$。

然后在满足等式的基础上调整。可以选定一行 i 和变量 x_i，让这一行的元素交替加、减 x_i，也就是 $a_{i,j} \leftarrow a_{i,j} + (-1)^j x_i$。类似地对列调整，选定一列 j 和变量 y_j，令 $a_{i,j} \leftarrow a_{i,j} + (-1)^i y_j$。只要确定 a 的第一行第一列就可以推出整个矩阵，所以这 $n+m$ 个变量已经足够调整出任何满足等式的矩阵。

约束是 $0 \le a_{i,j} + (-1)^j x_i + (-1)^i y_j \le 10^6$。凑一下差分约束的形式。令 $x_i \leftarrow (-1)^i x_i$，$y_j \leftarrow (-1)^{j+1} y_j$，约束转化为 $-a_{i,j} \le (-1)^{i+j}(x_i - y_j) \le 10^6 - a_{i,j}$。

时间复杂度是 $O(Tnm(n+m))$，可能卡常。

一般图最大独立集

求一般图最大独立集是 NP-hard 问题。本部分的时间复杂度均忽略了 $\mathrm{poly}(n)$ 因子，其中 n 是节点数量。

比暴力快的算法

设 $f(S)$ 表示点集 S 的导出子图的最大独立集大小。

当 $S=\varnothing$ 时，显然 $f(S)=0$。否则，设 $u=\max S$，有 $f(S)=\max\{f(S-\{u\}),f(S-\{u\}-N(u))+1\}$。

也就是说，最大独立集要么不包含 u，要么包含 u 且不包含所有和 u 相邻的节点。

比暴力快的算法

设 $f(S)$ 表示点集 S 的导出子图的最大独立集大小。

当 $S=\varnothing$ 时，显然 $f(S)=0$。否则，设 $u=\max S$，有 $f(S)=\max\{f(S-\{u\}),f(S-\{u\}-N(u))+1\}$。

也就是说，最大独立集要么不包含 u，要么包含 u 且不包含所有和 u 相邻的节点。

如果要求空间复杂度为关于 n 的多项式，时间复杂度是什么？记忆化搜索的时间复杂度呢？

局部记忆化

在 $S-\{u\}=S-\{u\}-N(u)$ 时（u 在导出子图中是孤立点），只需递归一次。

局部记忆化

在 $S-\{u\}=S-\{u\}-N(u)$ 时（u 在导出子图中是孤立点），只需递归一次。

记 $x=|S|$，若能发生两次递归，则 $|S-\{u\}-N(u)|\leq x-2$。设计算 $f(S)$ 的时间复杂度为 $T(x)$，则 $T(x)\leq T(x-1)+T(x-2)$。用 Fibonacci 数列估计 $T(n)=O(1.618^n)$。加上这个小优化已经比暴力快了（至少是理论上）!

指数空间复杂度的记忆化

$f(S)$ 总是向 $\max S$ 减小的方向递归。设节点编号为 $1 \sim n$。枚举 $1 \le i \le n$，被搜索到的满足 $\max S = i$ 的集合 S 数量受到两方面限制：

- 编号大于 i 的节点一定不在 S 中，节点 i 一定在 S 中，只有 2^{i-1} 种可能状态。
- S 由全集以某种方式删点得到，最坏情况下，删去 $i+1 \sim n$ 的每个节点时都有两种不同决策，结果数是 $O(2^{n-i})$。

指数空间复杂度的记忆化

$f(S)$ 总是向 $\max S$ 减小的方向递归。设节点编号为 $1 \sim n$。枚举 $1 \le i \le n$，被搜索到的满足 $\max S = i$ 的集合 S 数量受到两方面限制：

- 编号大于 i 的节点一定不在 S 中，节点 i 一定在 S 中，只有 2^{i-1} 种可能状态。
- S 由全集以某种方式删点得到，最坏情况下，删去 $i+1 \sim n$ 的每个节点时都有两种不同决策，结果数是 $O(2^{n-i})$。

将限制取 min 并求和，得到时间复杂度 $O(2^{\frac{n}{2}})$，大约就是 $O(1.414^n)$。

指数空间复杂度的记忆化

$f(S)$ 总是向 $\max S$ 减小的方向递归。设节点编号为 $1 \sim n$。枚举 $1 \le i \le n$，被搜索到的满足 $\max S = i$ 的集合 S 数量受到两方面限制：

- 编号大于 i 的节点一定不在 S 中，节点 i 一定在 S 中，只有 2^{i-1} 种可能状态。
- S 由全集以某种方式删点得到，最坏情况下，删去 $i+1 \sim n$ 的每个节点时都有两种不同决策，结果数是 $O(2^{n-i})$。

将限制取 min 并求和，得到时间复杂度 $O(2^{\frac{n}{2}})$，大约就是 $O(1.414^n)$。

卡满的构造：将节点 i 和节点 $n-i+1$ 连边。

能不能再给力一点啊

以上算法几乎遍历了所有独立集，略微修改就可以求最大权独立集、大小为 k 的独立集个数。接下来要利用最大独立集独有的性质。

能不能再给力一点啊

以上算法几乎遍历了所有独立集，略微修改就可以求最大权独立集、大小为 k 的独立集个数。接下来要利用最大独立集独有的性质。

设 u 在 S 的导出子图中度数为 $d(u)$。递归的时间复杂度 $T(x) = T(x-1) + T(x-d(u)-1)$，$d(u)$ 越大递归就越快。

能不能再给力一点啊

以上算法几乎遍历了所有独立集，略微修改就可以求最大权独立集、大小为 k 的独立集个数。接下来要利用最大独立集独有的性质。

设 u 在 S 的导出子图中度数为 $d(u)$。递归的时间复杂度 $T(x) = T(x-1) + T(x-d(u)-1)$，$d(u)$ 越大递归就越快。

- 若 $d(u) = 0$，则 u 显然属于最大独立集。$f(S) = f(S - \{u\}) + 1$。
- 若 $d(u) = 1$，设和 u 相邻的节点为 v，所有最大独立集方案中 u, v 肯定恰好出现一个，且所有出现 v 的情况换成 u 仍然合法。所以直接把 u 塞进最大独立集而删除 v 不劣。$f(S) = f(S - \{u, v\}) + 1$。

能不能再给力一点啊

- 若 $d(u)=2$，设和 u 相邻的节点为 v,w，所有最大独立集方案中 u,v,w 肯定至少出现一个。
  - 如果 v,w 相邻，则所有最大独立集方案中 u,v,w 恰好出现一个，直接把 u 塞进最大独立集而删除 v,w 不劣。$f(S)=f(S-\{u,v,w\})+1$。
  - 否则，如果把 u 塞进最大独立集是劣的，说明最大独立集同时有 v,w。$f(S)=\max\{f(S-\{u,v,w\})+1,f(S-\{u,v,w\}-N(v)-N(w))+2\}$。注意，由于和“局部记忆化”相同的原因，$T(x)\le T(x-3)+T(x-4)$。
- 否则 $d(u)\ge 3$，直接递归，$T(x)\le T(x-1)+T(x-4)$。

能不能再给力一点啊

- 若 $d(u)=2$，设和 u 相邻的节点为 v,w，所有最大独立集方案中 u,v,w 肯定至少出现一个。
  - 如果 v,w 相邻，则所有最大独立集方案中 u,v,w 恰好出现一个，直接把 u 塞进最大独立集而删除 v,w 不劣。$f(S)=f(S-\{u,v,w\})+1$。
  - 否则，如果把 u 塞进最大独立集是劣的，说明最大独立集同时有 v,w。$f(S)=\max\{f(S-\{u,v,w\})+1,f(S-\{u,v,w\}-N(v)-N(w))+2\}$。注意，由于和“局部记忆化”相同的原因，$T(x)\le T(x-3)+T(x-4)$。
- 否则 $d(u)\ge 3$，直接递归，$T(x)\le T(x-1)+T(x-4)$。

最后得到了时间复杂度 $T(n)=O(1.380^n)$，多项式空间复杂度的算法。可以在 $\max S$ 足够小的时候用数组（而不是哈希表）轻松实现记忆化，但复杂度未知。

能不能再给力一点啊

接下来不要求 $u = \max S$。

- 若存在 $d(u) \leq 2$，沿用上文的分类讨论，$T(x) \leq T(x-3) + T(x-4)$。
- 若存在 $d(u) \geq 4$，直接递归，$T(x) \leq T(x-1) + T(x-5)$。
- 否则这个图每个节点度数均为 3，任取一个节点删去后，都剩下 3 个 2 度点。$T(x) \leq T(x-4) + T(x-5) + T(x-4)$。

能不能再给力一点啊

接下来不要求 $u = \max S$。

- 若存在 $d(u) \leq 2$，沿用上文的分类讨论，$T(x) \leq T(x-3) + T(x-4)$。
- 若存在 $d(u) \geq 4$，直接递归，$T(x) \leq T(x-1) + T(x-5)$。
- 否则这个图每个节点度数均为 3，任取一个节点删去后，都剩下 3 个 2 度点。$T(x) \leq T(x-4) + T(x-5) + T(x-4)$。

时间复杂度 $T(n) = O(1.325^n)$。

Robson (2001) 讨论了 $d(u) \leq 9$ 的所有情况，得到了时间复杂度 $O(1.189^n)$ 的算法！

CF1578K Kingdom of Islands

给定一张 n 个节点的图，每个节点有颜色，节点之间有边当且仅当颜色不同。给定 k 对节点，它们之间有边当且仅当颜色相同。求最大团并给出构造。

$1 \leq n \leq 10^5$，$0 \leq k \leq 20$。

题解

求最大团就是求补图的最大独立集。

尝试缩小点数。如果两个节点颜色相同，且都没有特殊连边，则这两个节点是等效的，删去其中一个不会影响答案。如果一种颜色没有任何节点有特殊连边，则一定可以加进最大团中。

题解

求最大团就是求补图的最大独立集。

尝试缩小点数。如果两个节点颜色相同，且都没有特殊连边，则这两个节点是等效的，删去其中一个不会影响答案。如果一种颜色没有任何节点有特殊连边，则一定可以加进最大团中。

对至多 $4k$ 个节点的导出子图跑最大独立集。$O(1.414^{4k})$ 和 $O(1.380^{4k})$ 都可以通过。