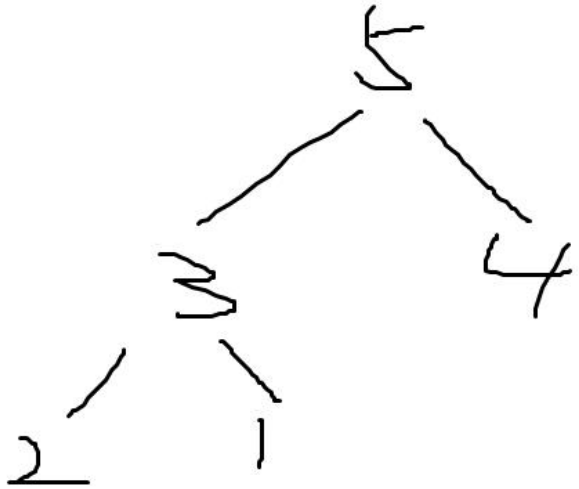
堆和并查集

堆

- 堆是一棵完全二叉树。
- 对于大根堆，每个节点的权值比它的两个子节点（若存在）都大。
- 对于小根堆，每个节点的权值比它的两个子节点（若存在）都小。
- 一般来说给节点编号为1至n，i 号节点的子节点为 $2i$ 和 $2i+1$ 。



- 加入新元素?
- 逐层向上判断是否需要与父节点交换即可。
- 大根堆:
- void add(int v)
- {
- 	t[++n]=v;
- 	从n开始自底向上:
- 		若比父节点更大则向上交换
- }

- 删除堆顶元素?
- 先将 1 号节点与 n 号节点交换，再从 1 号点逐层向下，每次判断是否需要与更大的子节点交换即可。
- 大根堆:
- void pop()
- {
- 	swap(t[1],t[n]);
- 	n--;
- 	从1开始自顶向下:
- 		若小于子节点中更大的那个则与之交换
- }

堆排序

- 堆排序就是逐个插入 n 个元素，再弹出 n 次栈顶元素的过程。

实际应用

- 一般使用STL的priority_queue（优先队列）。
- 该容器默认为大根堆。
- 如何变为小根堆?
- 1.
- priority_queue<typename,vector<typename>,greater<typename> >
- 2.
- 若对自定义结构体建堆，重载小于运算符时按着大于的逻辑来即可。

- 能否删除堆中任意元素（而非堆顶元素）？
- 若知道删除元素在堆中的位置，仿照删除堆顶即可。
- 若不知道，无法快速找到对应位置。
- 解决办法：另开一堆存放被删除元素。

```
struct Heap
{
    priority_queue<int> A,B;//B存放删除元素
    void push(int x){A.push(x);}
    void del(int x){B.push(x);}
    int top(){while(A.top()==B.top())A.pop(),B.pop();return A.top();}
    void pop(){while(A.top()==B.top())A.pop(),B.pop();A.pop();}
};
```

- 如何支持合并两个堆?
- 每次选择较小的堆，将其元素一一取出，并一一插入较大的堆中。
- 复杂度?
- 若元素总数为 n，则每个元素被这样插入一次，其所在的堆的大小会至少翻一倍，因此每个元素至多被插入 $O(logn)$ 次，复杂度就是 $O(nlog^2n)$ 。
- 可并堆（左偏树）暂不介绍，有兴趣的可以了解一下。

- 习题1
- 你需要经营一个只卖一种货物的商店，初始时没有库存，共有 n 天，事先知道了第 i 天上午进货量为 a_i ，以及第 i 天中午会有一名购买量为 b_i 的顾客。求最多能满足多少顾客的需求。
- $n \leq 10^5$

- 按天数顺序考虑每个顾客，能满足就满足，否则考虑反悔之前某一个已经满足的顾客。
- 显然是选择之前满足过的顾客中需求最大的一个，在当天顾客需求更小时选择反悔。
- 大根堆维护即可。

- bzoj2802

- 习题2
- 有 n 个人，每个人有一台电脑，第 i 个人的电脑的初始电量为 a_i ，每过一分钟会消耗电量 b_i 。
- 你有一个充电器，每分钟只能给一个人的电脑充电量为 x 的电，你需要最小化这个 x ，能让前 k 分钟所有人的电量都不小于0。
- 注意电量没有容量限制，可以任意地充电。
- $n \leq 10^5$

- 显然可以二分答案，考虑对于固定的 x 判断合法性。
- 第一分钟应该给哪个人充电?
- 应当选择当前最快没电的人，也就是 $\left\lfloor \frac{a_i}{b_i} \right\rfloor + 1$ 最小的人。
- 用 c_i 记录当前第 i 个人被充电的次数。
- 用小根堆维护每个人当前没电的时间，从1到 k 枚举时间，每次取出堆顶，若堆顶的没电时间小于当前时间，说明二分的 x 不合法；否则，设堆顶是第 i 个人，弹出堆顶， c_i++ ，将 $\left\lfloor \frac{a_i+c_i\times x}{b_i} \right\rfloor + 1$ 入堆。

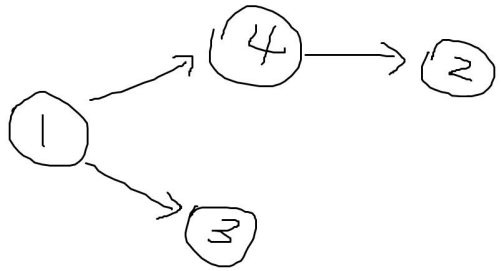
- cf1132D

- 习题3
- 一张有向无环图，求出一个拓扑序，使得：
- 1.1号点在拓扑序中的位置尽可能靠前。
- 2.满足1的前提下，2号点在拓扑序中的位置尽可能靠前。
- 3.满足1，2的前提下，3号点在拓扑序中的位置尽可能靠前。
- …以此类推

- $n \leq 10^5$
- 3 2
- 1 3
- 1 2

- 1 2 3

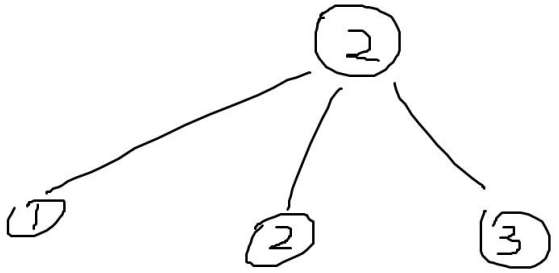
- 考虑拓扑序的求法，维护一个入度为0的队列，每次取出一个元素并更新其出边指向的点。
- 每次从队列中取最小元素?
- 显然不对，反例如图所示。



- 最优为 $\{1,4,2,3\}$ 而非 $\{1,3,4,2\}$ 。
- 按拓扑序从前往后来看，某一时刻能选择的两个点，并不一定是选较小的更优。
- 按拓扑序从后往前来看，根据题意，我们应当让更小的点出现地尽可能晚，每次应当取尽可能大的点。
- 因此，我们对原图的反图跑拓扑排序，用大根堆维护队列，每次取最大的点放入拓扑序，最后翻转输出即可。

- bzoj4010

- 习题4
- 给定一棵 n 个节点的有根树， i 号节点有点权 v_i 。你需要求出一个尽可能大的点集，使得对于点集中的任意两点 x, y ，若 x 是 y 的祖先，则 $v_i > v_j$ 。
- $n \leq 10^5$



- 该图点上的值为 v_i 。
- 若选择根节点，总共只能选两个节点。
- 若不选根节点，则可以选所有三个子节点。

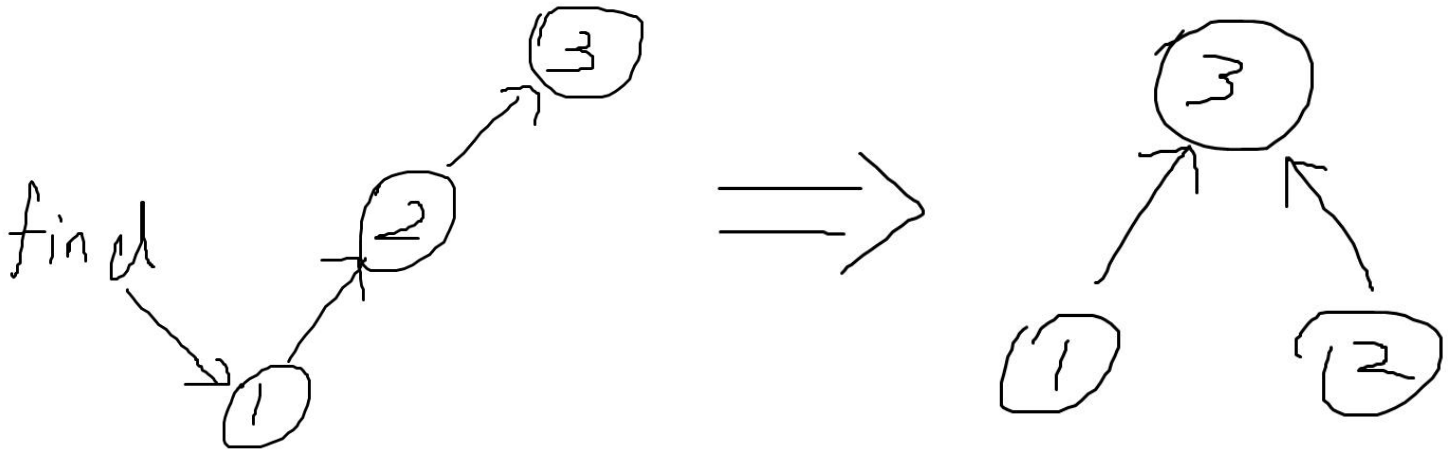
- 提示：考虑树为一条链的情形。

- 若树是一条链，则等价于求一个最长上升子序列。
- 求法众多，选哪一种?
- 设 f_i 表示上升子序列长度为 i 时，末尾元素的最小值，每次加入元素时，在 f 序列中二分并替换。
- 扩展到树的情形?
- f_i 表示在当前子树选择的点集大小为 i 时，点集中最大的点权 v 的最小值。
- 合并节点 u 的所有子树的 f ?
- 子树之间没有影响，因此把所有子树的 f 中所有元素都扔到一起排序即可，最后把 v_u 在其中二分并替换。
- 把每个 f_i 想成是每次多选一个节点的点权，显然是按点权从小到大选点，更好理解上述合并方式的正确性。
- 用multiset存储 f ，并使用启发式合并。 $O(nlog^2n)$
- bzoj4919

并查集

- n 个点，两种操作：
- 1.加一条双向边 (x, y) 。
- 2.询问两点 u, v 是否连通。

- int find(int x){return fa[x]==x?x:fa[x]=find(fa[x]);}
- void link(int x,int y){fa[find(x)]=find(y);}
- $find(x)$ 称为 x 所在连通块的代表元素，或者是该连通块的根。
- 两个节点在同一个连通块中，也就是它们的根相同。
- 路径压缩：把find路径上的点都直接挂到根上，从而减小fa结构的深度。
- 复杂度为 $O(n\alpha(n))$， $\alpha(n)$ 可认为是一个非常小的常数，图示如下：



最小生成树

- 求一个带边权的无向连通图的最小生成树。
- 按边权排序逐个加边，若当前边的两个端点未连通则加边。
- 用并查集来维护连通性。

- 习题5
- 有 n 个人， m 个冲突关系 x_i, y_i, v_i 表示若第 x_i 个人与第 y_i 个人分到了一组，他们之间会发生冲突值为 v_i 的冲突。
- 你需要把所有人分为两组，使得冲突值最大的冲突最小。
- $n, m \leq 10^5$

- 按冲突值从大到小尽可能化解冲突，也就是分到不同的组。
- 若 a 与 b 不同组， b 与 c 不同组，则 a 与 c 必须同组。
- 设 x 所在连通块表示当前必须与 x 同组的人。
- 设 $x+n$ 所在连通块表示当前必须与 x 不同组的人。
- 对于每个冲突 x, y ，若 x 与 y 连通，或 $x+n$ 与 $y+n$ 连通，则该冲突无法避免，否则 $link(x, y+n), link(y, x+n)$ 。

- NOIP2010

- 习题6
- 有 n 个人，每个人至少要 a_i 分，但每个人的分不能相同（且都要是整数），因此只能给某些人额外加一些分数。
- 求一个使总额外加分最少的方案。
- $n \leq 10^5, a_i \leq 10^{18}$

- 3
- 5 1 1

- 5 1 2

- 这题做法非常多，这里讲一个并查集做法。
- 并查集除了普通地维护图的连通性之外，也可以维护序列，从某位置开始的满足某些要求的左边界和右边界。
- 在这题中，设 $fa[x]$ 表示从 x 开始，向右的连续段的右端点。
- 比如若3，4，5都被用过，6没有被用过，则 $find(3) = find(4) = find(5) = 5$ 。
- 当前考虑第 i 个人，若 a_i 未被选择，就选择 a_i ，否则选择 $find(a_i) + 1$ 。
- 也就是，每个人贪心地取这个连续段更右一位的位置即可。
- 选完 x 之后要连接 $x - 1$ 与 x ， x 与 $x + 1$ （若存在的话）。
- 用map存fa。

- cf379C

- 习题7
- n 个点， m 条边，第 i 条边连接 u_i, v_i ，并有一个区间 $[l_i, r_i]$ 。找一条从 1 号点到 n 号点的路径，使得路径上的所有边的区间的交集尽可能大。
- $n, m \leq 3000$

- 4 4
- 1 2 1 10
- 2 4 3 5
- 1 3 1 5
- 2 4 2 7
- 选择第1条边和第4条边，交集为 $[1,10] \cap [2,7] = [2,7]$ 最大。

- 枚举最终交集的左端点 L，对于所有 $l_i \le L$ 的边，按 r_i 从大到小加入并查集，等到 $find(1) = find(n)$ 时停止，此时就找到了该左端点对应的最大右端点。

- cf366D

- 习题8
- 你与你的竞争对手去铺设一个 n 个节点的网络。
- 你的竞争对手已经给出了他的 p 条边，每条边都定好了边权。
- 你现在只有你的 q 条边的端点，边权还未确定。
- 你需要给你的 q 条边都确定一个边权，使得对于 n 个点和所有的 $p+q$ 条边，存在一个最小生成树包含你的所有 q 条边。在此基础上，你的 q 条边的边权和应该尽可能大。
- $n, p, q \leq 10^5$

- 首先，自己的 q 条边不能有环，否则一定有边选不上。
- 这样，自己的 q 条边会将 n 个点连成一个森林（若干棵树）。
- 按边权从小到大考虑竞争对手的所有 p 条边，若不连接同一棵树，就将这两棵树连接，否则若有边连接同一棵树，考虑这条边的端点在树上的路径，若树上的路径中有某条自己的边的边权比竞争对手这条边的边权更大，那么那条树边就无法出现在最小生成树里了。因此，对于这条路径上的所有树边，其边权都至多为竞争对手这条边的边权。
- 由于按边权从小到大枚举，相当于每次给路径上未赋权的边赋一个权。直接用并查集来维护，初始时不进行路径压缩，每次路径赋权后再路径压缩到LCA上。

- cf1023F

可撤销并查集

- n 个点，三种操作：
- 1.加一条双向边 (x, y) 。
- 2.询问两点 u, v 是否连通。
- 3.删除上一条加入的边。

- 朴素的并查集为何不支持撤销?
- 路径压缩整体复杂度优秀，但单次复杂度可达 $O(n)$ （压缩一条链），在此刻反复加边与撤销就会被卡掉。
- 放弃路径压缩？这样每次find需要沿着fa一直找到根。
- 如何保证路径的深度?
- 额外记录每个连通块的最大深度，连接时由深度小的连向深度大的，连接深度相同的连通块时，注意让最大深度+1。
- 这样就能保证最大深度是 $O(logn)$ 级别的。
- 如何支持撤销?
- 用一个栈记录对fa和dep的所有改动，发生撤销时，通过栈顶信息来得知需要还原的内容。
- 复杂度降为 $O(nlogn)$ 。

```
inline void link(int x,int y)
{
	x=find(x);y=find(y);
	if(x==y)return;
	if(dep[x]>dep[y])swap(x,y);//按深度连边
	if(dep[x]==dep[y])dep[y]++,stk[++top]=-y;//将dep修改压栈，负数表示dep变化
	fa[x]=y;stk[++top]=x;//将fa修改压栈，正数表示fa变化
}
inline void revoke(int x)//x是需要恢复到的栈顶位置
{
	while(top>x)
	{
		if(stk[top]<0)dep[-stk[top]]--;//撤销dep修改
		else fa[stk[top]]=stk[top];//撤销fa修改
		top--;
	}
}
```

- 习题9
- n 个点，支持三种操作：
- 1.加边。
- 2.删边。
- 3.查询某两点是否连通。
- $n \leq 10^5$

- 提示：将操作离线处理

- 删除的边不一定是最近加入的边了，无法方便地撤销。
- 将所有操作离线之后，若一条边在 L 时刻加入， R 时刻删除，那么这条边在时间上就在 $[L, R-1]$ 这个线段中都存在。
- 把每条边按其时间线段打到线段树上的 $O(logn)$ 个节点中，每个线段树节点存一个vector，保存被打到这个点上的所有边。
- 考虑第 t 个时刻会出现的所有边，就是从线段树 $[t, t]$ 这个叶子一直往上直到根节点的所有线段树节点中存的所有边。
- 从线段树的根节点开始深度优先遍历这棵线段树，访问一个节点时加入它存的所有边，离开一个节点时撤销它存的所有边。访问到叶子节点时就可以知道对应时刻的连通性了。
- $O(nlog^2 n)$
- bzoj2049

- 习题10
- 一个长为 n 的序列，给出 m 个限制，第 i 个限制为序列的区间 $[l_i, r_i]$ 的和为 v_i ，求是否存在满足所有限制的序列。
- $n, m \leq 1000$
- 5 3
- 1 5 100
- 3 5 50
- 1 2 51
- NO
- 由 [1,2] 和为51， [3,5] 和为50可推出 [1,5] 和为101，与第一条限制矛盾。

- 设 sum 为该序列的前缀和，则限制被修改为：
- $sum[r_i] - sum[l_i - 1] = v_i$
- 用并查集维护被限制联系的下标集合，设：
- $d[x] = sum[x] - sum[fa[x]]$
- 若 x, y 不属于同一个集合，就合并并更新 d 。
- 若 x, y 在并查集中有同一个根 z ，我们可以先用 d 求出 $d[x] - d[z], d[y] - d[z]$ ，相减就得到 $d[x] - d[y]$ ，从而判断合法性。
- 当然，本题中只询问全局的合法性，完全可以先把所有边加上，每个连通块任选一点开始dfs，此时 $d[x]$ 表示 x 与该点的差，并借此判断合法性。前述做法只是带权并查集的展示。

- bzoj1202

- 上题中带权并查集的find：
- int find(int x)
- {
- 	if(fa[x]==x)return x;
- 	int f=fa[x];
- 	fa[x]=find(f);
- 	d[x]+=d[f];//更新到根的权值和
- 	return fa[x];
- }
- 什么情况下可以使用带权并查集？
- 由a与b的权值关系，b与c的权值关系可以推出a与c的权值关系。
- 权值关系有类似可加性的性质。

- 习题11
- 一张 n 个点的无向图，给出全部 m 条边的两个端点以及出现时间和消失时间，求对于所有的 T 个时刻，该图是否为二分图。
- 一张无向图是二分图，等价于该图中不存在奇环。
- $n, m, T \leq 10^5$
- 3 3 3
- 1 2 0 2
- 2 3 0 3
- 1 3 1 2
- 0时刻时只有 $(1,2), (2,3)$ 两条边，无奇环。
- 1时刻有 $(1,2), (2,3), (1,3)$ 三条边构成的奇环。

- 在之前的习题中我们已经学会了将边按出现时间打到线段树上的操作了，因此现在只需考虑如何判断是否为二分图。
- 一张无向图是二分图，等价于该图中不存在奇环。
- 考虑带权并查集， $d[x]$ 表示 x 到 $fa[x]$ 的路径长度的奇偶性， $dist(x)$ 表示 x 到根的路径长度的奇偶性。
- 加边 (u,v) 时，若两点不连通，设其根分别为 x,y ，且根据深度应由 x 连向 y ，则 $d[x]=dist(u)\text{^}dist(v)\text{^}1$ ，也就是从 x 走到 y 需要沿着 $x\to u\to v\to y$ 路径。
- 若两点连通，若 $dist(u)\text{^}dist(v)=0$ ，则 u 到 v 路径长为偶数，加上边 (u,v) 形成奇环，不合法；反之则合法，且 (u,v) 这条边可被当前 u 到 v 的这条长为奇数的路径替换（长度都是奇数），因此可以直接忽略这条边。

- bzoj4025

- 习题12
- 一个 $n \times m$ 的矩阵 A，有两种可能的操作：
- 1.选择一行，将该行的所有元素+1或-1。
- 2.选择一列，将该列的所有元素+1或-1。
- 现在告诉你所有操作结束后 k 个格子的最终值，求是否存在一个满足条件的操作序列。
- $n, m, k \leq 1000$

- 等价于第 i 行有一个 a_i，第 j 列有一个 b_j，最终矩阵中第 i 行第 j 列的元素就是 $a_i + b_j$。
- 考虑一个条件：第 p 行第 q 列的元素最终值为 r，也就是：
- $a_p + b_q = r$
- 设 $v_i = a_i, v_{j+n} = -b_j$，则条件改写为：
- $v_p - v_{q+n} = r$
- 与之前的bzoj1202完全相同，带权并查集或dfs解决。

- bzoj4500

- 习题13
- 一张 n 个点， m 条边的无向图，每条边有一个边权。
- 求出一棵该图的生成树，使得最小的没有出现在其边权集合中的自然数尽可能小，输出这个最小值。
- $n, m \leq 10^5$
- 3 3
- 1 2 0
- 2 3 1
- 2 3 2
- 1
- 选择连接 $(1,2)$， 边权为0的边和连接 $(2,3)$， 边权为2的边。

- 先考虑0是否能成为答案，即是否存在一棵不含0边权的生成树。
- 把所有非0边加入，看整个图是否连通即可。
- 若0不是答案，则考虑1是否能成为答案。
- 此时可把0边全部加入，大于1的边也全部加入，看图是否连通。
- 继续讨论可以发现，我们要求的就是对于每一个边权 i，判断用所有边权不为 i 的边能不能使整个图连通。
- 整个图连通就是连通块数量为1，可在并查集发生合并时维护。
- 分治+并查集，按边权分治，先把右半边的所有边加入，分治左半边，然后撤销并把左半边的所有边加入，分治右半边。这样每当分治到叶子时，其余所有边权的边都加入了并查集。

- 习题14
- 一张 n 个点， m 条边的无向图，每条边有端点 u_i, v_i ， 长度 d_i ，高度 h_i 。
- 多次询问，每次给你一个 x 和一个 v ，从节点 x 出发，开车经过任意的 $h_i \leq v$ 的边到达一个节点 y ，然后从节点 y 开始，步行前往1号节点。最小化步行总长度。
- $n, m \leq 10^5$

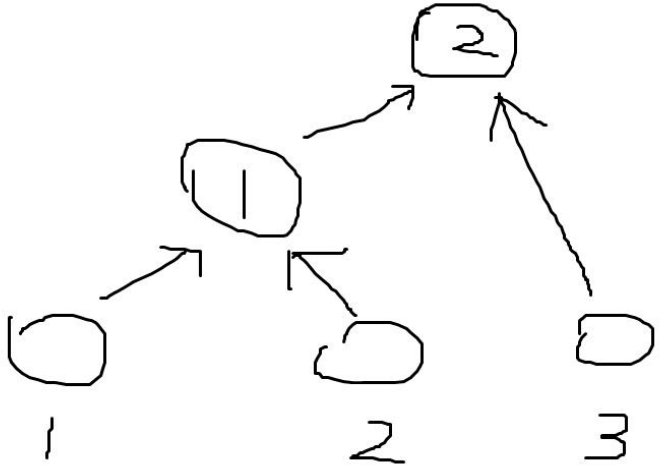
- 提示：将询问离线

- 题目分为两部分。
- 第一部分为在由 $h_i \leq v$ 的边连成的， x 所在的连通块中找一个点 y ，第二部分为计算从 y 到1的最短路。
- 先计算1号点出发到 u 的单源最短路 d_u 。
- 将所有询问按 v 从小到大排序，不断地加入 $h_i \leq v$ 的边，并维护连通块内的 d_u 的最小值。

- 习题15
- 题面同上一题，但强制要求在线回答询问。

- 一张 n 个点， m 条边的无向图，每条边有端点 u_i, v_i ，长度 d_i ，高度 h_i 。
- 多次询问，每次给你一个 x 和一个 v ，从节点 x 出发，开车经过任意的 $h_i \leq v$ 的边到达一个节点 y ，然后从节点 y 开始，步行前往1号节点。最小化步行总长度。
- $n, m \leq 10^5$

- 可持久化并查集？有更简单的做法么？
- 考虑一棵这样的树，n 个节点都作为叶子节点，在按 h_i 加边时，每次若合并了两个集合，就新建一个点权为 h_i 的点，表示这两个集合的根节点的父亲。
- 该图表示了以 $h_i = 1$ 连接 $(1,2)$，再以 $h_i = 2$ 连接 $(2,3)$ 后该树的形状。



- 可以发现，任取该树的某个子树，设子
- 树的根的点权为 p，这个子树的所有叶
- 子就是一个由 $h_i \le p$ 组成的连通块。
- 对于每一个询问 (x, v)，从叶子 x 开始，
- 倍增找到不超过 v 的最浅的祖先，其子
- 树即为所求的连通块，预处理子树最小值即可。
- uoj393