最小权路径集问题

【解题报告】

关键词：斯坦纳树、DP 套 DP、最短路优化 DP

将这些路径分为若干个部分，使得每个部分都相交，不同部分不相交。那么每个部分的所有路径的端点都连通。

记 $f(i,S)$ 表示使点 i 与集合 S 中每一个点都连通的最小代价。其中 $S\subseteq\{1,2,...,n\}$ 。

那么有状态转移：

（1）$f(i,S)\leftarrow f(i,T)+f(i,S-T)$ ，其中 $T\subset S$ 。

这一转移表示：将 S 划分为若干个不相交子集，如果 x 和 $y(x,y\le k)$ 不属于同一个子集，那么 i 到 x 、i 到 y 的路径就不重合。

我们考虑像背包问题或者树形 DP 一样，一个个地加入集合，就能够推出上述转移。

（2）$f(i,S)\leftarrow f(j,S)+w(i,j)$

这一转移表示：一切从 i 出发到集合 S 中任一点的路径都两两重合。首先，任意两点间最多有一条路径是被计算了答案的，不然一定不优。

所以，我们的第一步一定是重合的，也就得到上述转移。这个转移由于可以相互转移，要使用 SPFA（当然也可以用其他最短路算法）来转移。

这只是一部分的答案，最后再做一次 DP 将这些合并起来就好了。

铃铛计数问题

【解题报告】

题面分析：

在一棵有根树上进行单点修改，区间查询。

算法分析：

首先应该想到暴力的解法，就是预处理父节点，每次修改把会影响到的点都改了，求和就直接区间求和。

期望得分：$0pts$

优化 1 ：

区间求和很麻烦，我们可以想到用分块来维护 s，查询的时候整块就直接加上去，不满的块暴力查询。

但是问题来了：**如何实现修改？**

考虑到题目意思是单点修改，而修改时会对祖先产生影响，因故我们可以预处理出对应关系。

记：$f_{i,j}$ 表示点 i 对第 j 块产生影响的权值和。

记：sum_i 表示第 i 块的和，那么在修改的时候就可以直接用 tag_i 来记录对块 i 的影响。

于是乎，区间查询的时候就可以用 $sum_i + tag_i$ 计算出答案了。

对于整块的做法是如此的，但是问题又来了：**不满的块要怎么暴力？**

不满的块单点暴力固然好算，不过因为考虑到 s_i 的实际意义，在进行单点修改的时候会很麻烦。

那要怎么修改会比较方便呢？

优化 2 ：

借鉴优化１，不难想出可以对 w_i 进行分块，那么 s_i 就可以表示为部分 w_j 的和。

可是编号是毫无意义的，我们不难发现，一棵树上节点的 dfs 序是有意义的，

不妨求出每个节点的 dfs 序，然后记 $w'_i = w_{dfni}$。

于是 s_i 就可以表示为一段连续 w'_j 的和。

对 w'_j 进行分块 ,那么在求不满块的 s_i 的时候就可以简化为求 w'_j 的区间和。

复杂度为 $O(n^2)$ 。

期望得分： $50pts$

优化 3 ：

因为 s_i 上分块是区间修改，区间查询，复杂度均为 $O(\sqrt{n})$,

而对 w'_j 分块是单点修改，区间查询，复杂度分别为 $O(1)$ 和 $O(\sqrt{n})$,

明显会 TLE ，那么有没有一种做法可以使对 w'_j 的分块优化成区间修改，单点查询呢？

答案显然是有的。

我们可以考虑对 w'_j 的后缀和进行分块！(PS:为什么是后缀和请读者自己思考)

当 w'_j 发生变动的时候，会对区间 $[1, j]$ 产生影响。

于是我们记 $sum'_j = \sum_{t=j}^{n} w'_t$ ，记 tag'_i 表示第 i 块的影响。

那么在修改的时候，对整块直接修改tag'_i，不满的块暴力修改就行了。

求和的时候就直接用前缀和减一下就可以了，记住加上tag 。

期望得分：$100pts$

越野赛车问题

【解题报告】

首先对于 $n,m \le 20$ 的数据，随便乱搜都能过。

然后是对于 $l_i = 1$ 的数据，也就是说最小允许的速度都是 1，只要考虑 r_i 造成的影响。

所以可以想到按照边的 r_i 降序排序，同时把询问**离线**也按照 v_i 降序排序，每次只需要把所有 r 比当前询问 v 大的边加入某个数据结构维护，然后更新答案即可。

由于这里维护的是连通性，所以很自然的想到**并查集**。但是要求的答案是最长链的长度，所以需要对并查集中的每一个连通子树维护其中**最长链的两个端点**，设为 $\{x,y\}$ 。

合并两个并查集的时候，假设另一个并查集最长链的两个端点为 $\{s,t\}$ ，那么合并后子树的最长链只有六种情况：$\{x,y\},\{s,t\},\{x,s\},\{x,t\},\{y,s\},\{y,t\}$ ，求两点距离的时候用**树链剖分** LCA 转换一下。

这样时间复杂度就是 LCA + 并查集 + std::sort = $O(n\log n)$ 。

对于所有的数据，l_i 不一定都为 1 ，也就是说必须考虑 l_i 造成的影响。若此时还是按照之前的做法，那就需要带权并查集支持任意撤销一条边的连接，这是非常不现实的。

换一种思考方法，把区间 $[l_i,r_i]$ 看作**时间区间**，那么一条边将在 l_i 时刻出现，并在 r_i 时刻后消失，询问 v 就是询问 v 时刻的答案。

同时注意到 $l_i,r_i,v_i \le n \le 70000$ ，所以想到**线段树分治**。

线段树分治其实就是**按照时间分治**，因为分治的区间 $[l,mid],(mid,r]$ 极其类似于线段树分治的区间形式（几乎是一样的），所以可以用分治的方法遍历线段树，同时合并（**利用**）线段树上**维护的信息**进行答案的更新。

对于此题来说，对 v 进行分治，分治区间 $[l,r]$ 维护速度**在** $[l,r]$ **范围**时的合法的边。显然不能对每个这样的区间（总共不超过 $4n$ 个）都存下所有的合法的边，这时候线段树的思想就派上用场了。

线段树查询的复杂度只有 $\log n$ ，因为若区间 $[l,r]$ 已经被询问区间完全包含，那么就**不用**继续往下寻找，直接利用这个区间点的信息更新答案即可。同理，线段树分治也是如此。若一条边在速度为区间 $[l,r]$ 的范围内已经合法，那么就不用对于 $[l,r]$ 内的其余所有子区间都添加这条边了，因为只有分治到 $[l,r]$ 时需要添加这条边的信息，之后分治进的子区间就再也不需要了。这样一条边就最多只会被 $O(\log n)$ 个区间所添加，时间和空间复杂度都是可以接受的。

现在还剩最后一个问题，分治时需要在返回之前撤销在这一层中并查集添加的边。发现只要简单的按**栈序撤销**即可，所以并查集不能有路径压缩，写一个按秩合并（启发式合并），还原时记得把 f 数组，$size$ 或 $rank$ ，维护的最长链的两个端点，所有连通子树中最长链最大长度（答案） 全部还原！

这样时间复杂度就是 LCA + 并查集 + 线段树分治 = $O(n\log n)$ 。