【题解】2023 牛客 NOIP 赛前集训营-提高组（第二场）

T1-集合

对从 1 到 n 中每个数可选可不选，共有 2^n-1个非空子集。

现在考虑每个非空子集的和的贡献，子集和的取值范围是 $[1,\frac{n*(n+1)}{2}]$，可以 $O(n^3)$ 求每一种子集和的方案数，然后枚举子集和 x，我们又知道子集和的方案数 y，那么根据题意全部相乘就是 x^y，需要用快速幂解决。

由于 n 较大的时候 dp 数组会超过 $long\,long$ 存储范围，而 dp 数组是不能求余 998244353 的（它存储的内容的含义是 x^y 中的幂次 y）注意到子集和与模数 998244353 一定是互质的，所以考虑欧拉定理或者费马小定理，对 dp 数组求余 $\varphi(p)=p-1$。

T2-出租

考虑什么时候是无解的。

当出现任意一段区间 $[l,r]$ 的租户满足它们人数的和比 $k*(r-l+1+d)$ 还多的时候，说明无论如何也无法给 $[l,r]$ 中的所有人都安排房间。

我们对这个式子进行作差，得到 $\sum_l^r(val-k)>k*d$，val 就是当前位置的人数。可以这样理解：$[l,r]$ 这些人可以被分配到 $[l,r+d]$ 这些位置，每个位置 k 个人，那么总共就能够装下 $k*(r-l+1+d)$ 个人。将这个式子拆分成 $k*(r-l+1)+k*d$，其中左边 $k*(r-l+1)$ 是变量（因为我们不确定 l,r 的值，对本题来说，每一个 l,r 都需要满足要求），左边的值和 $[l,r]$ 的已有租户人数作差，看看差值是否超过 $k*d$，如果超过，则说明无法满足。

综上：用线段树维护最大子段和，然后和 $k*d$ 比大小即可。

T3-连通块

80pt

枚举每个连通块在原树上深度最高的点

考虑一定包含深度的点最高为 x 的连通块，约定对于每个结点，其前戳为该点的 dfs 序，后戳为其子树中最大的 dfs 序，按 dfs 序标号，在 i 号点上有两种决策，要么选择该点转到 $i+1$，要么割掉以 i 为根的子树，转到 i 的后戳 $+1$。

对每个点建立入点和出点，在它们之间连接权值为点权的边，将上述转移建图，i 的出点连向 $i+1$ 的入点，i 的入点连向 i 的后戳 $+1$ 的入点，权值都为 0。这样每一个连通块都对应了图上的一条从 x 出发的路径。

对于每个限制，约定 $dfn[u]<dfn[v]$ 特判掉两点连通时中间有其它点的情况，直接将断开 u 的出边向外连的边，从 u 到当前根的所有结点的儿子中，所有没有限制的结点 x，从 u 的出点向 x 的出点连一条权值为 val_x的边。

由于树是随机生成的，所以总结点数是 $nlogn$ 级别。

100 pt

$f_{i,j}$表示表示子树的根为 i 且 dfs 序最后一个的是 j 的最大值

```
#include<bits/stdc++.h>
#define rep(i,x,y) for(int i=x; i<=y; ++i)
#define repd(i,x,y) for(int i=x; i>=y; --i)
#define mid ((l+r)>>1)
#define lch (rt<<1)
#define rch (rt<<1|1)
#define pb push_back

using namespace std;
typedef long long LL;
const int N=100005,M=52;
const LL INF=1e18;
bool vis[N],mp[M][M];
int n,a[N],k,m,id[N];
LL ans,f[N][M];
struct D {
    int u,v;
```

```
} dat[M];
vector <int> vt[N];

int getint() {
    char ch;
    int f=1;
    while(!isdigit(ch=getchar())) if(ch=='-') f=-1;
    int x=ch-48;
    while(isdigit(ch=getchar())) x=x*10+ch-48;
    return x*f;
}

void dfs(int x) {
    rep(i,0,k) f[x][i]=-INF;
    f[x][id[x]]=a[x];
    for(auto v:vt[x]) {
        dfs(v);
        LL mx=-INF;
        rep(i,0,k) if(!mp[i][id[v]]) mx=max(mx,f[x][i]);
        rep(i,0,k) f[x][i]=max(f[x][i],f[v][i]+mx);
    }
    rep(i,0,k) ans=max(ans,f[x][i]);
}
int main() {
    cin >> n >> m;
    rep(i,1,n) scanf("%d", a + i);
    ans = a[1];
    rep(i, 2, n) ans = max(ans, (LL)a[i]);
    if(ans<=0) return printf("%lld\n",ans),0;
    rep(i,1,n) {
        int x, k;
        scanf("%d", &k);
        while(k--) {
            scanf("%d", &x);
            vt[i].pb(x);
        }
    }
    rep(i,1,m) {
        int u, v;
        scanf("%d%d", &u, &v);
        dat[i]=(D) {
            u,v
        };
        vis[u]=vis[v]=1;
```

```
    }
    rep(i,1,n) if(vis[i]) id[i]=++k;
    memset(vis,0,sizeof(vis));
    rep(i,1,m) {
        int u = id[dat[i].u], v = id[dat[i].v];
        mp[u][v] = mp[v][u] = 1;
    }
    dfs(1);
    printf("%lld\n",ans);
}
```

T4-跳棋

对于 subtask1-2：

- 直接枚举每个 ? 位置是否有棋子，然后记忆化搜索。

对于 subtask3：

- [因为可以 OEIS 到所以大概可以推出什么神秘的组合意义](http://oeis.org/A025565)。

对于 subtask4：

- 经过大眼观察法，你发现如果有两个贴贴的 11，它们可以一起去往相邻任意空的位置，于是根据这一个变化规律大概可以推出只有一段 1 的变化情况。

对于 subtask5 $O(n^3)$：

- 经过超大眼观察法，你可以发现 011 变成 110 虽然是一个 1 跳过去，但是其实可以看作 11 和 0 换位置。

- 于是就直接 $f_{i,j,k,0/1}$ 表示已经填了前 i 位，有 j 个 11，k 个 0，且 i 是否可以在后面接一个 1 变成 11。

- 最后对于每种 $f_{\mathrm{n,i,j}}$，把 0/1 两种情况加起来乘上 $\binom{i+j}{i}$ 就好了。

- 解释一下$\binom{i+j}{i}$是怎么来的。

  - 你发现一个 0 是无法跨过长度为奇数的 1 的连续段的。

  - 所以对于序列中的任意两个 0， 中间的 1 的奇数段的数量是一定的。

  - 所以答案是$\binom{i+j}{i}$。