提高组Day1试题 第一组 题解

1273 旅行计划

解

直接说具体做法，做法中蕴含的意义，请大家自行理解。

1、对整棵树做DFS，并记录节点深度。

2、按照节点深度及编号对节点排序，深度越大排序越靠前，深度相同时，编号小的排在前面。

3、按照排好序的节点顺序，逐个处理节点，从叶子结点向上走到父节点，并对已走过的节点标注为已访问。直到走到某个已访问的节点或者根节点后停止，记录走过的步数。

4、按照走过的步数及编号，对节点做第二次排序，得到的结果即为答案。

排序可以使用基数排序，因此复杂度为$O(n)$。

标准代码

C++:

```
#include <bits/stdc++.h>
using namespace std;

int N, K, L, t;
int parent[50000];
int length[50000];
vector<vector<int> > tree;
vector<vector<int> > res;
vector<vector<int> > tmp;
int main()
{
    cin >> N >> K;
    tree.resize(N); tmp.resize(N); res.resize(N);
    for(int i = 1; i < N; ++i) {
        scanf("%d", &t);
        tree[t].push_back(i);
        tree[i].push_back(t);
    }
    vector<int> st;
    st.push_back(K);
    parent[K] = K;
    length[K] = 0;
    for(; !st.empty(); ) {
        t = st.back();
        st.pop_back();
        for(int i = 0; i < tree[t].size(); ++i) {
            if(tree[t][i] != parent[t]) {
                parent[tree[t][i]] = t;
                length[tree[t][i]] = length[t] + 1;
```

```
                st.push_back(tree[t][i]);
            }
        }
    }
    for(int i = 0; i < N; ++i) {
        if(tree[i].size() == 1) {
            res[length[i]].push_back(i);
        }
    }
    for(int i = N - 1; i > 0; --i) {
        for(int j = 0; j < res[i].size(); ++j) {
            int len = 0;
            int cur = res[i][j];
            for(; length[cur] != 0; ++len) {
                length[cur] = 0;
                cur = parent[cur];
            }
            length[res[i][j]] = len;
        }
        res[i].clear();
    }
    for(int i = 0; i < N; ++i) {
        if(tree[i].size() == 1) {
            res[length[i]].push_back(i);
        }
    }
    printf("%d\n", K);
    for(int i = N - 1; i > 0; --i) {
        for(int j = 0; j < res[i].size(); ++j) {
            printf("%d\n", res[i][j]);
        }
    }
    return 0;
}
```

1354 选数字

解

类似于背包的DP，以乘积为状态。先把等选数字里面不是K约数的去掉。然后找出K的约数，进行离散化。然后dp[i][j]表示前i个数字乘积为j的状态。$Dp[i+1][j*a[i+1]]+=dp[i][j]$.

$Dp[i+1][j]+=dp[i][j]$;

总的复杂度是$O(n*d(k)*log(d(k)))$

$D(k)$表示k的因子数目。多一个log是因为离散化了，对应下标的时候要二分查找。

标准代码

Visual C++:

```
#include <iostream>
#include<time.h>
#include<stdio.h>
```

```
#include<algorithm>
#include<math.h>
#include<vector>
#include<string.h>
#include<assert.h>
using namespace std;
typedef __int64 lld;
const int BIT = 100;
const int MAX = 10020;
const int MOD = 1000000007;
int d[MAX], c;
int a[MAX];
int dp[MAX];
int splite(int d[], int n)
{
    int i, len = 0;
    for (i = 1; i*i <= n; i++)
    {
        if (n%i == 0)
        {
            d[len++] = i;
            if (i*i != n)d[len++] = n / i;
        }
    }
    sort(d, d + len);
    return len;
}
int bin(lld aim, int low, int high)
{
    lld ret = -1;
    while (low <= high)
    {
        int mid = low + high >> 1;
        if (aim == d[mid])return mid;
        if (d[mid]<aim)
        {
            low = mid + 1;
        }
        else high = mid - 1;
    }
    return ret;
}
int main()
{
    int i, j, k;
    int T, n,ioflag;

    ioflag=scanf("%d", &T);
    assert(ioflag == 1);
    while (T--)
    {
        ioflag=scanf("%d%d", &n, &k);
        assert(ioflag == 2);
        assert(1 <= n&&n <= 1000);
        assert(2 <= k&&k <= 100000000);
        c = 0;
        for (i = 0; i<n; i++)
        {
```

```
            ioflag=scanf("%d", &a[c]);
            assert(ioflag == 1);
            assert(1 <= a[c] && a[c] <= k);
            if (k%a[c] == 0)c++;
        }
        n = c;
        //for (i = 0; i < n; i++)printf("a[%d]=%d\n", i, a[i]);
        c = splite(d, k);
        for (i = 0; i < c; i++)dp[i] = 0;
        dp[0] = 1;
        //for (i = 0; i < c; i++)printf("d[%d]=%d\n", i, d[i]);
        for (i = 0; i<n; i++)
        {
            for (j = c - 1; j >= 0; j--)
            {
                if (dp[j] == 0)continue;
                int id = bin((lld)d[j] * (lld)a[i], 0, c - 1);
                if (id != -1)
                {
                    dp[id] += dp[j];
                    if (dp[id] >= MOD)dp[id] -= MOD;
                }
            }
            //printf("i=%d a[%d]=%d\n", i, i,a[i]);
            //for (j = 0; j < c; j++)printf("dp[%d]=%d\n", j, dp[j]);
        }
        //for (i = 0; i < c; i++)printf("dp[%d]=%d\n", i, dp[i]);

        printf("%d\n", dp[c - 1]);
    }
    ioflag = scanf("%d", &T);
    assert(ioflag != 1);
    return 0;

}
```

2553 双重祖先

解

我们可以通过dfs序对第一棵树进行编号，得到每个节点本身的编号以及子树的编号范围（L_i，R_i）。

用树状数组来维护后面的计算。

在DFS处理第二棵树时，我们每处理完一个节点，就将该节点在第一棵树中的对应编号设为1，处理完所有子节点后，查询编号区间(L_i, R_i)的区间和，即当前子树下，有多少个节点在树1中也是当前节点的子孙节点。维护计数即可。

复杂度$nlog(n)$。

标准代码

C++:

```
#include <bits/stdc++.h>
```

```
using namespace std;
typedef long long LL;
#define N 500000 + 10

int t,n;
vector<int> a[N], b[N];
int cnt=1, num2[N], out[N];
int mk[N], d[2 * N];
int res[N];
int ww=0;
void DFS2(int u, int prev)
    {
        num2[u]=cnt++;
        for (int i=0; i < b[u].size(); i++)
            {
                int v = b[u][i];
                if (v == prev) continue;
                if (num2[v] == 0)
                    DFS2(v, u);
            }
        out[u]=cnt++;
    }
void update(int x, int val)
{
        while (x <= 2 * n)
            d[x]+=val, x+=x & (-x);
}
int get(int x)
{
        int rs=0;
        while (x > 0)
            rs+=d[x], x-=x & (-x);
        return rs;
}
void DFS1(int u, int prev)
    {
        res[u]=get(num2[u]);
        update(num2[u],1);
        update(out[u],-1);

        for (int i=0; i < a[u].size(); i++)
            {
                int v=a[u][i];
                if (v == prev) continue;
                DFS1(v, u);
            }
        update(num2[u],-1);
        update(out[u],1);
    }
int main()
{
    scanf("%d", &n);
    for (int i=1; i <= n; i++)
    {
        a[i].clear();
        b[i].clear();
        num2[i]=out[i] =d[i] =res[i]=0;
        cnt = 1;
```

```
    }
    for (int i=1; i < n; i++)
    {
        int u,v;
        scanf("%d%d", &u, &v);
        a[u].push_back(v);
        a[v].push_back(u);
    }
    for (int i=1; i < n; i++)
    {
        int u,v;
        scanf("%d%d", &u, &v);
        b[u].push_back(v);
        b[v].push_back(u);
    }
    DFS2(1, 0);
    DFS1(1, 0);
    long long ans = 0;
    for (int i=1; i <= n; i++) {
        ans += res[i];
    }
    printf("%lld\n", ans);
    return 0;
}
```