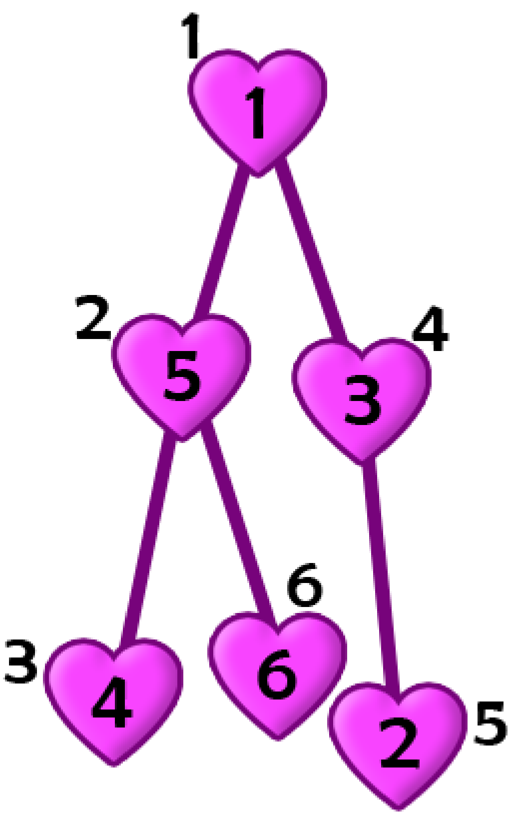
[hanging hearts]

chenjiuri_hanginghearts_dfdf

```
给定一个树结构。
标记位1....n;
在上面自主的填一个数字x，其中该数字只能选自【1,,,n】的排列之中。
并且每一个数字只能够选择一次:
每一次操作是如下的过程:
1.选择一个没有联系到其它子节点的节点x;
2.将该卡片上的额数字x追加到字符串的尾部。
3.如果x! =1.且父节点上的数字大于其数字，交换两个数字。
4.将x删掉。
```

20mins

- 第一点问题该怎么去分配数字到每一个节点之上?
- 第二个问题是：如果已经确定了分配的顺序之后，又怎么去确定一个最终的字符串的序列?
- 分析结论如下：
  - 发现，可以总是可以构造出一个最长子序列为两个互不相交，只在根节点相聚的每棵子树的最大深度之和。



- 这里就是一个深度和为4的子树。

- 尝试证明结论。
  - 是否存在一个比之更大的情况?
  - 该结论本身是错误的。由于，简单找一个特例就知道了。同时我们也有关注更小、处的分叉情况。

---

solve

- 关于结论，对于每一颗子树，按照特殊的分配形式，将前若干部分分配到子树之上，这样本质相同，同时使得两颗子树得到的最长子序列可以拼接。
- 子树的最长上升子序列的方式有两种。
  - 树的最高高度。
  - 几颗子树一起拼接起来，但是根并没有作为一个贡献。所以就是子树的最大上升子序列之和。
- 状态转移方程为： $f_x = max(h_x, \sum f_{sons})$
  - 实现上，树形dp，dfs即可。

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

void MAIN();
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr), cout.tie(nullptr);
    MAIN();
}
typedef long long ll;
const int maxn = 2e5 + 10;
//------code-----٩(ˊωˋ*)و ------靓仔代码-----٩(ˊωˋ*)و ----talk is cheap , show me the code--------

class tree_list
{
    struct node
    {
        int no, to;
    };

public:
    node e[maxn << 1];
    int head[maxn];
    int tot;
    void add(int x, int y)
    {
        e[++tot].no = y;
```

```cpp
            e[tot].to = head[x];
            head[x] = tot;
        }
};
tree_list t1;
int f[maxn];
int h[maxn];

void dfs(int now)
{
    for (int x = t1.head[now]; x; x = t1.e[x].to)
    {
        dfs(t1.e[x].no);
        f[now] += f[t1.e[x].no];
        h[now] = max(h[now], h[t1.e[x].no]);
    }
    h[now]++;
    f[now] = max(h[now], f[now]);
}

void MAIN()
{
    int n;
    cin >> n;
    for (int i = 2; i <= n; i++)
    {
        int y;
        cin >> y;
        t1.add(y, i);
    }
    dfs(1);
    cout << f[1] << '\n';
}
```

生长思考：

- 对于别人来说，很容易发现，但是对自己来说有一定的难度。
- 总思路：
  - 构造方案。
  - 转化为动态规划问题。（不同规模之间的子问题之间有迁移关系。）