算法分析与设计
第三章

- 14181
- 1034
- 1140
- 1219
- 1310
- 1426
- 7766
- 1504
- 2002
- 1170
- 1490

这次的评分标准

- 通过任意一题并写出实验报告，即可有8分（即使是写的14181题）
- 通过任意两题并写出一题的实验报告，即可有9分。其中14181题不计入实验报告的评分内
- 通过任意三题并写出至少两题的实验报告，即可有9.5分，其中14181题不计入实验报告的评分内，另外需要通过(1140,1219)两题的其中一题
- 通过任意四题并写出至少三题的实验报告即可有满分，其中并写出至少两题的实验报告，即可有满分，其中14181题不计入实验报告以及题数的评分内，另外报告中需包含(1140,1219)两题中的其中一题。

这次的评分标准

- 当然，你可以选择无视上面的所有步骤，进而选择完成下面的步骤来获得满分：
- 完成1140,1219的其中一题
- 另外独立完成4426题以及其实验报告，本次实验不负责这道题有关的任何解答
- 另外需交4426的代码来查重

Soj 14181 Flying Safely

- 题意：给出一个连通的图，问遍历所有的点最少需要经过多少边（一条边可以多次经过，而在答案中只算一次）
- 题解：由于保证是连通的，所以答案是点数减一。(by poetry)

Soj1034 Forest

- 题意：
- 给一个n个点，m条有向边的图
- 问这个图是不是森林，如果是输出这个森林的最大深度和最大宽度
- 森林的定义为：一个有向图，没有指向同一个节点的边也没有重边
- 对于入度为0的点，被称为为根，它们处于第0层
- 对于一条有向边，u->v，如果u处于第k层，那么v处于第k+1层
- 最大深度指的是存在点最深一层
- 最大宽度是指处于同一层的最多点数
- 限制：1<=n<=100, 0<=m<=100,m<=n*n

Soj1034 Forest

- 同样是在树上做统计，这次我们需要计算每个节点的深度depth
- 我们可以使用一个深搜（dfs）来完成这项工作

Soj1034 Forest

- 不是森林的有两种情况：
- 第一种可以在深搜的时候得到
- 这一种是在深搜时，多次访问同一节点
- 这个是指向同一结点的其中一种情况
- 重边的情况同样可以在这一步避免
- 这样就不需要特殊判断重边了

Soj1034 Forest

```
int n, m;
int maxDepth, maxWidth;
int in[maxn];
bool mark[maxn];
int width[maxn];
bool dfs(int u, int depth)
{
    mark[u] = true;
    width[depth]++;
    maxDepth = max(maxDepth, depth);
    for (size_t i = 0; i < G[u].size(); i++)
    {
        int v = G[u][i];
        if (mark[v])
            return false;
        else if (!dfs(v, depth + 1))
            return false;
    }
    return true;
}
```

Soj1034 Forest

- 另一种情况则产生于存在环

```
maxDepth = maxWidth = 0;
for (int i = 0; i < n; i++) if (in[i] == 0)
    if (!dfs(i, 0))
        return false;
for (int i = 0; i < n; i++) if (!mark[i])
    return false;
for (int i = 0; i < n; i++)
    maxWidth = max(maxWidth, width[i]);
```

Soj1034 Forest

- 主过程：

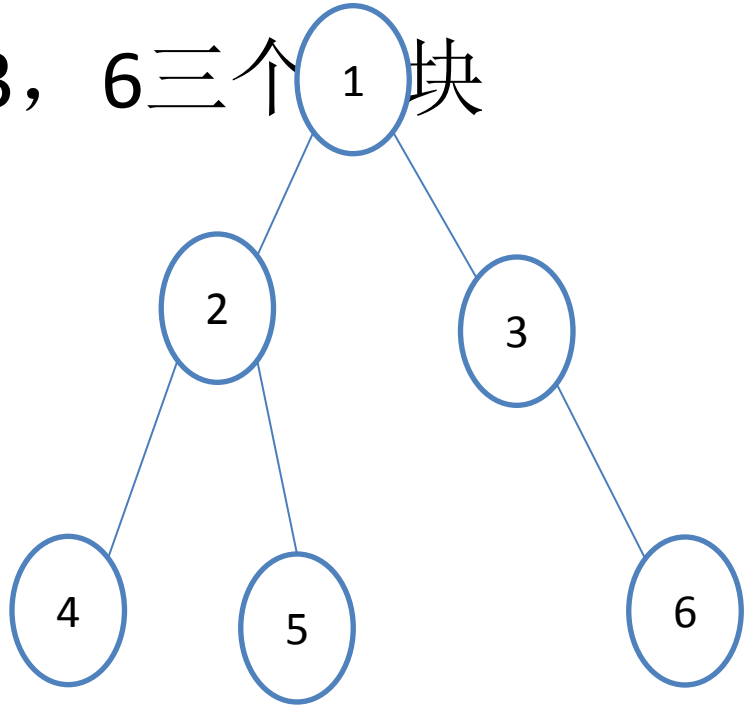
```
bool forest()
{
    for (int i = 0; i < n; i++)
    {
        G[i].clear();
        mark[i] = false;
        width[i] = 0;
        in[i] = 0;
    }
    for (int i = 0; i < m; i++)
    {
        int u, v;
        scanf("%d %d", &u, &v);
        u--, v--;
        G[u].push_back(v);
        in[v]++;
    }
    maxDepth = maxWidth = 0;
    for (int i = 0; i < n; i++) if (in[i] == 0)
        if (!dfs(i, 0))
            return false;
    for (int i = 0; i < n; i++) if (!mark[i])
        return false;
    for (int i = 0; i < n; i++)
        maxWidth = max(maxWidth, width[i]);
    printf("%d %d\n", maxDepth, maxWidth);
    return true;
}
```

Soj 1140 国王的遗产

- 题意：
- 有一个国王拥有一个n个金块组成的树，在他死后由他的k个儿子轮流分金块。
- 每个人可以选择一条边将它断开，然后选择金块数量少的那一块，如果金块数量相同，则选择剩余编号小的金块所在的那一块。
- 约束：n<=3*10^4,k<=10^2

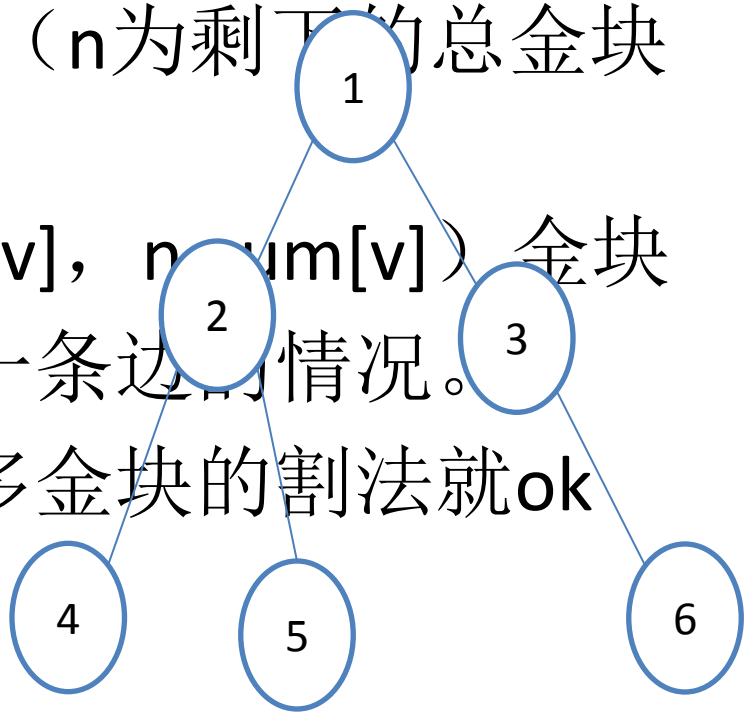
Soj 1140 国王的遗产

- 题目已经说了每个王子的选择都是贪心的，那么我们只需要模拟这个过程就可以了。
- 如右图所示，现在可以断开1和2
- 之间的边，然后取得1，3，6三个块



Soj 1140 国王的遗产

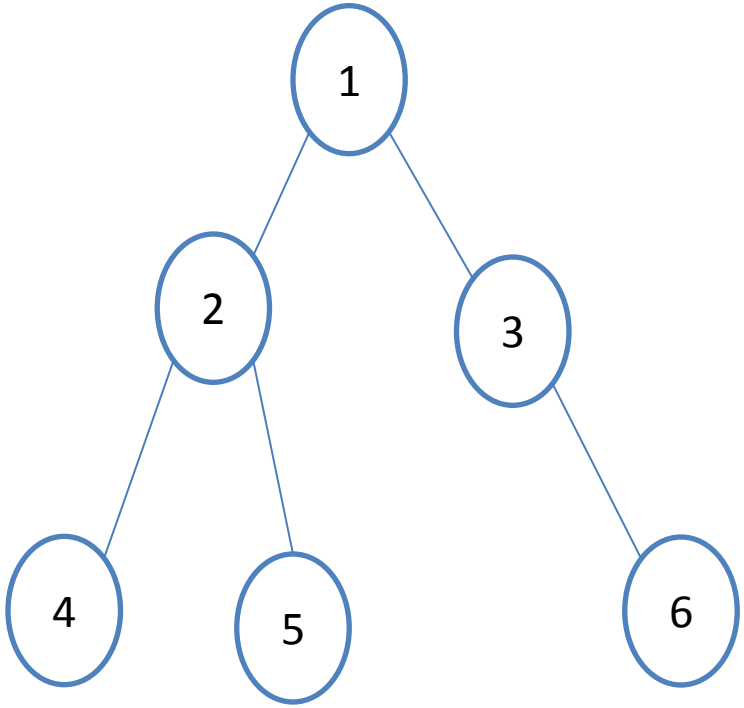
- 另sum[u]，表示以当前最小标号的金块作为根时，以u为根的子树的大小
- 那么考虑割掉一条边（u，v），那么会分为sum[v]，与n-sum[v]两部分（n为剩下的总金块数）。
- 显然此时会取得min（sum[v]，n-sum[v]）金块
- 所有我们只需要枚举断掉一条边的情况。
- 然后选择其中能够获得最多金块的割法就ok



Soj 1140 国王的遗产

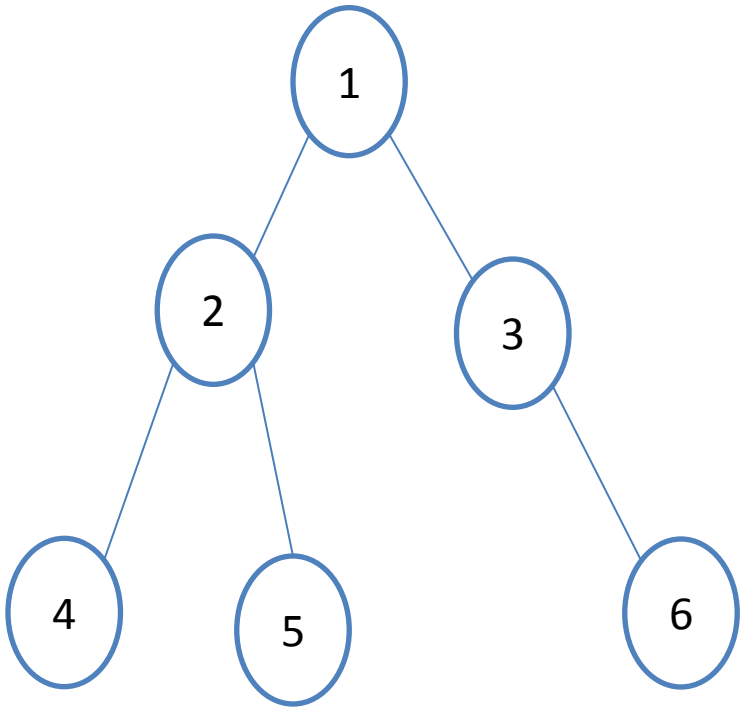
- 剩下一个问题就是计算sum[u]，这个只需要在树上做统计就可以了

```
int sum[maxn];
vector<int> G[maxn];
void count_sum(int u, int fa)
{
    sum[u] = 1;
    for (int i = 0; i < G[u].size(); i++)
    {
        int v = G[u][i];
        if (v == fa) continue;
        count_sum(v);
        sum[u] += sum[v];
    }
}
```



Soj 1140 国王的遗产

- 这道题其实是树的分治的一个基础（树的边分治）



Soj 1219 新红黑树

- 题意：
- 一棵树由红枝和黑枝组成的树，A和B轮流砍树，A只砍红枝，B只砍
  黑枝
- 砍枝后不与根相连的枝都去掉。每个树枝上有权值，砍掉的枝的权值
  加到自己的分数上
- A想使A-B之差越高越好，B想它越低越好。在最佳策略下A-B之差
- 限制：
- 树枝数不超过20

Soj 1219 新红黑树

- 分析：
- 树枝树不超过20，这个条件是个重要的突破口，这意味着我们可以使用O(2^n)级别的的算法
- 那我们不妨考虑使用（mask，round）表示一个博弈状态，表示当前树枝状态为mask，现在轮到round这个人砍树
- 这个基础上，我们就得到了一种记忆化搜索的做法

Soj 1219 新红黑树

- 解法：
- 首先我们需要预处理砍掉一个树枝后，跟着消失的树枝有哪些
- Cut[i]表示砍掉第i个树枝一起

合

```
const int maxn = 20;
int n;
int f[1<<maxn][2];
bool cal[1<<maxn][2];
int cut[maxn];
vector< pair<int, int> > G[maxn];
int c[maxn], w[maxn];

int dfs(int u, int fa)
{
    int mask = 0;
    for (size_t i = 0; i < G[u].size(); i++)
    {
        int v = G[u][i].first, id = G[u][i].second;
        if (v == fa) continue;
        int nmask = dfs(v, u);
        cut[id] = 1 << id | nmask;
        mask |= cut[id];
    }
    return mask;
}
```

Soj 1219 新红黑树

- 解法：
- 接着就是记忆化搜索的主过程

```
int dp(int mask, int round)
{
    if (cal[mask][round])
        return f[mask][round];
    cal[mask][round] = true;
    if (mask == 0)
        return f[mask][round] = 0;
    bool has = false;
    for (int i = 0; i < n; i++) if (mask >> i & 1 && round == c[i])
    {
        int nex = dp(mask & ~cut[i], round ^ 1) + (round ? w[i]: -w[i]);
        if (!has) f[mask][round] = nex;
        else
        {
            if (round)
                f[mask][round] = max(f[mask][round], nex);
            else
                f[mask][round] = min(f[mask][round], nex);
        }
        has = true;
    }
    if (!has) f[mask][round] = dp(mask, round ^ 1);
    return f[mask][round];
}
```

Soj 1219 新红黑树

- 完整解法：
- http://soj.sysu.edu.cn/viewsource.php?sid=4389911

Soj 1310 Right-Heavy Tree

- 题意：
- 给你一个输入序列，依次插入一个二叉搜索树中（初始为空树）
- 要输出它的前序遍历、中序遍历和后序遍历
- 限制：
- 节点的数目不超过200000

Soj 1310 Right-Heavy Tree

- 朴素解法：
- 直接实现二叉搜索树的插入操作以及树的遍历

- 由于数据比较水，实现比较好可以水过这道题

Soj 1426 Phone List

- 题意：
- 给你n个电话号码，问你是否存在一个电话号码是另一个电话号码的前缀
- 限制：
- 1<=n<=10000
- 电话号码长度不超过10

Soj 1426 Phone List

- 解法：
- 其实是一个简单的Trie（字典树）的应用

Soj 1426 Phone List

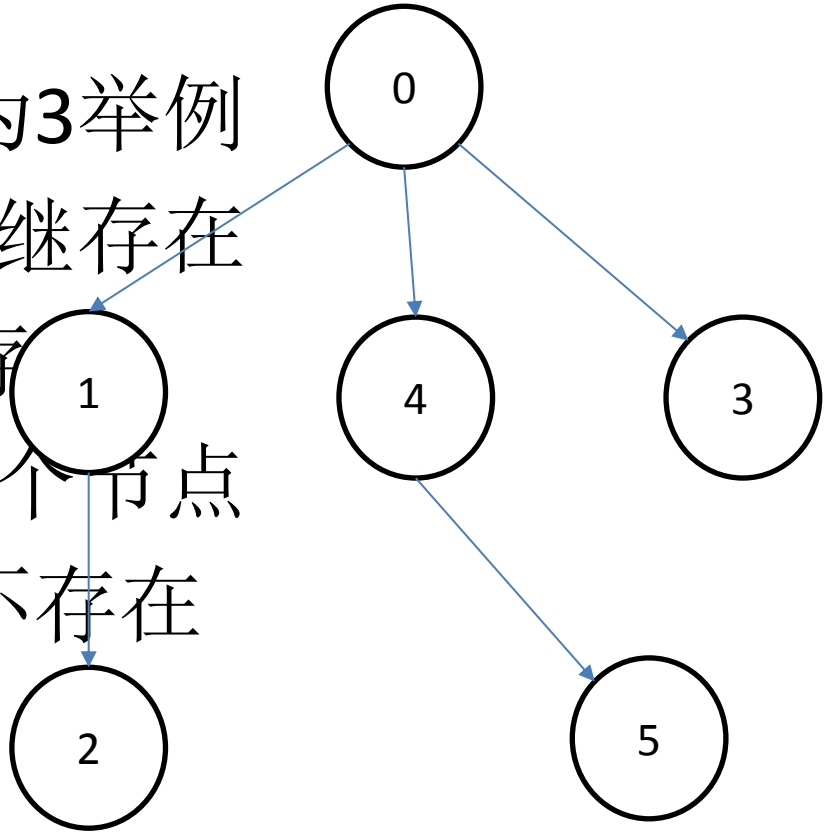
- 字典树：
- 这是一个数组版本

```
int tot;
bool isEnd[maxn];
int nxt[maxn][10];
```

- 字典树的每个节点都有一
- 对于节点i可以保存其相关信息
- 如isEnd[i]标记这个节点是不是一个单词的结尾
- nxt[i][]是必须的保存的是这个节点的后继 nxt[i][j]表示第j个后继，注意的是这里的后继可以不存在，nxt[i][]数组的大小取决于字符集的大小

Soj 1426 Phone List

- 字典树:
- 以字符集大小为3举例
- 节点1第一个后继存在
- 而第二个后继存
- 我们用0标记这个节点
- 的某一个后继不存在



Soj 1426 Phone List

- 字典树:
- 插入操作
- 注释掉的为题目相关

```
void insert(char *s)
{
    int cur = 0;
    for (; *s; s++)
    {
        int idx = *s - '0';
        if (!nxt[cur][idx])
        {
            isEnd[tot] = false;
            memset(nxt[tot], 0, sizeof(nxt[tot]));
            nxt[cur][idx] = tot++;
        }
        cur = nxt[cur][idx];
//      if (isEnd[cur])
//          ok = false;
    }
    isEnd[cur] = true;
//  for (int i = 0; i < 10; i++)
//      if (nxt[cur][i])
//          ok = false;
}
```

Soj 1426 Phone List

- 解法：
- 这道题要判断是否存在一个字符串是另一个字符串的前缀
- 当我们依次插入这些字符串时
- 第一种是在之前已经插入了现在插入的字符串的前缀，这种情况在我们顺着字典树插入的时候，必然会碰到某个单词的结尾；
- 第二种情况是这个字符串是之前插入的字符串的前缀，这种情况则会发现把当前字符串插入完成之后，最后的节点仍然存在后继

soj 7766 Dark roads

- 题意：n个点，m条边，给出每个边的权值，起点和终点，求出其最小生成树后，所删除的边的权值。
- 题解：Kruskal算法求出最小生成树权值，并将总的权值减去该权值即可。(by poetry)

Soj 1504 Slim Span

- 题意：一个无向图，求出生成树后需满足，该生成树中最大与最小边的差值最小。
- 题解：将边按照权值从小到大排序，然后枚举从哪个位置开始将前面的边删除，剩下的边用来尝试构建最小生成树，如果成功则更新答案。(by poetry)

Soj 2002 Feeding Time

- 题意：在w*h的矩阵中，求出最大的8连通块的大小，其中'.'是块，'*'是障碍。
- 题解：运用bfs，将每个'.'点所有相邻的没有标记的'.'点标记，并加入到队列里，之后再将该点弹出。如果开始搜索时队列为空，则停止搜索，并返回该连通块的大小。将所有这样的连通块取个最大值即可。(by poetry)

soj 1170 Countdown

- 题意，给出一个家族谱，然后求出前3个拥有d层儿子最多的人(例如d=1为儿女辈，d=2为孙子辈之类)，然后如果拥有儿子个数为0的则不考虑，也不会出现在结果里。如果在第三个数量上拥有并列者，则将所有的并列者输出。

soj 1170 Countdown

- 题解：首先名字的话，可以用map为每个名字分配一个编号，方便数组的访问。
- 遍历每个点，如果用dp(n,d)表示点n的d层孩子数，那么dp(n,d) = sum(dp(s,d-1)),其中s是n的直接儿子。
- 最后将所有的结果整理，输出答案即可 (by poetry)

1490 Tree Grafting

- 题意：给出一个字符串，根据这个字符串建树的过程如下：

u:回到当前节点的父亲节点

d:新开一个节点，作为当前节点的儿子，并到那个新的节点上

现在需要求当前树的高度，以及将其变成二叉树之后，二叉树的高度。

1490 Tree Grafting

- 变二叉树的方法如下：
- 对每个节点，将其第一个儿子做为其左儿子，将其下一个兄弟作为右儿子

1490 Tree Grafting

- 题解：可以直接根据题目中的条件，建立一颗树，然后按照规则再将其转化为一颗二叉树，分别统计两者的高度即可。
- 思考：有没有不建树，直接做的方法？(by poetry)