树

1.树的搜索--

> order traverse

思路1:bfs(要建树的数据结构)

思路2:dfs(要Node结构体)

```
struct sq {
        int index;
        int data;
};
void order(int i,int j) {
        if (node[i].lchild == -1 && node[i].rchild == -1) {
                in.push_back(i);
                return;
        }
        if (node[i].lchild != -1) {
                level.push_back({2 * j,node[i].lchild});
                order(node[i].lchild,2 * j);
        }
        in.push_back(i);
        if (node[i].rchild != -1) {
                level.push_back({2 * j+1,node[i].rchild});
                order(node[i].rchild,2 * j+1);
        }

}
bool cmp(sq a,sq b) {
        return a.index<b.index;
}
```

> 搜符合要求路径(不需要Node结构体)

```
const int maxn = 1010;
vector<int> g[maxn];//已知结点关系
void dfs(int x, int dp) {//depth运用
    path[depth] = w[x];
    if (g[x].size() == 0) {
        ll temp = 0;
        for (int i = 0; i <= depth; i++) temp += path[i];
        if (temp == s) {
            for (int i = 0; i <= depth; i++) ans[len].push_back(path[i]);
            len++;
        }
        return;
    }
```

```
    for (int i = 0; i < g[x].size(); i++) {
        dfs(g[x][i], dp + 1);
    }
}
```

2.空间问题

方法1:vector.resize(k)+vector本身也可以用下标(√)

方法2:int child[100]+int k=0;

3.查找vector中某值的lower uper--PAT1043

试比较以下和for循环+i--,会有for第一值退出时多减了 若是递增序列，不如直接找比他小的下一个，比他大的上一个

```
if(!isMirror) {
        while(i <= tail && pre[root] > pre[i]) i++;
        while(j > root && pre[root] <= pre[j]) j--;
} else {
        while(i <= tail && pre[root] <= pre[i]) i++;
        while(j > root && pre[root] > pre[j]) j--;
}
```

4.Node形参问题->段错误探讨

```
struct Node {
        vector<int> child;
        double sp = 0;
};
void dfs(int x,int dp) {
        int i;
        if(v[x].child.size()!=0) {
                for (i = 0; i<v[x].child.size(); ++i)
                if (dp+1 > pw.size()-1)
                        pw.push_back(pow(r, dp+1));
                        dfs(v[x].child[i],dp+1);
        }
        else sum += 1.0*v[x].sp*p*pw[dp];
}
```

如果将dfs第一个参数写为Node x会爆空间,更不谈存几个double了,直接用节点图或者层分布图vector g[maxn];

```
void dfs(int x, int depth) {
    if (g[x].size() == 0) {
        ans += (dat[x] * pow(1 + r, depth));
        return;
```

```
    }
    for (int i = 0; i < g[x].size(); i++) {
        dfs(g[x][i], depth + 1);
    }
}
```

段错误总结：

> (1)越界访问

数组开小了导致指针指向了为开辟的数组区域，出现了越界访问多层**for**循环中内层循环本来打算写**j**或者**k**，却因为习惯或忘记误写成了外层循环的变量**i**或**j**，导致数组访问**i**或**j**下标的时候发**生**了越界

> (2)栈空间和堆空间

大数组在**main**函数**里面**的话是存储在栈**里**，**而**栈空间是在进程创建时初始化，有固定的**大小**，**一**般为**几十KB**，所以太**大**的数组会耗光栈空间。**而**全局变量占**用**的堆空间，堆空间中的内存是按需分配，**自**由**增长**的，可以**非**常**大**，**比**如**32**位的系统中可以**大**到**4GB**。将**大**数组放在全局变量中能避免栈溢出

5.并查集

```
int findf(int v){
    if(v==father[v])//注意点1
        return v;
        else
        {
            int f = findf(father[v]);//注意点2，不然那无法结束
            father[v] = f;
            return f;
        }
}
void uni(int a,int b){
    int fa = findf(a);
    int fb = findf(b);
    if(fa!=fb)
        father[fa] = fb;
}
```

6.字符串模板

```
int str2num(char *a)
{
    return 26 * 26 * (a[0]-'A') + 26 * (a[1]-'A') + (a[2]-'A');
}
string num2str(int num){
    string a(3,0);
    a[0] = num / (26 * 26)+'A';
    a[1] = num / 26 % 26 + 'A';
```

```
        a[2] = num % 26 + 'A';
        return a;
    }
```