

点分治

Centroid Decomposition

Idea: 树上统计问题中，可分治地统计——先统计经过树根的路径，再统计各个的子树的答案。每次选择重心作为树的根节点，以保证时间复杂度。根据题目的不同，统计的方法也不同（即 `calc` 函数不同），而特别地，一些统计问题需要容斥或减去重复统计的答案。

Complexity: $O(n \lg n)$

ATT: 每一次 `getRoot` 前记得初始化。

Code:

```
1  int root, sum, mxson[N], size[N];
2  bool vis[N];
3  void getRoot(int x, int f){
4      mxson[x] = 0, size[x] = 1;
5      for(int i = head[x]; i; i = edge[i].nxt){
6          if(edge[i].to == f || vis[edge[i].to]) continue;
7          getRoot(edge[i].to, x);
8          size[x] += size[edge[i].to];
9          mxson[x] = max(mxson[x], size[edge[i].to]);
10     }
11     mxson[x] = max(mxson[x], sum - size[x]);
12     if(mxson[root] > mxson[x]) root = x;
13 }
14
15 void calc(int x){
16     ...; //根据题目写相应的统计函数
17 }
18
19 void solve(int x){
20     calc(x, 1); // 1 是为容斥方便，依题而定
21     vis[x] = 1;
22     for(int i = head[x]; i; i = edge[i].nxt){
23         if(vis[edge[i].to]) continue;
24         calc(edge[i].to, -1); // 容斥原理，依题确定统计是否需要容斥
25         root = 0, sum = size[edge[i].to], mxson[0] = INF; // 初始化
26         getRoot(edge[i].to, 0);
27         solve(root);
28     }
```

```
29 }
30
31 int main(){
32     ...;
33     root = 0, sum = n, mxson[0] = INF; // 每一次 getRoot 记得初始化
34     getRoot(1, 0);
35     solve(root);
36     ...;
37 }
```