基环树

基环树是一个n个点n条边的图，比树多出现一个环，因此称为基环树，也叫做环套树。

基环树

- 找环
  - 对于无向图
  - 对于有向图
- 习题
  - [ZJOI2008]骑士(基环树上DP)
  - [IOI2008] Island(基环树直径)
- 参考资料

找环

对于无向图

通过拓扑排序可以找出环上的所有点。

```
void topsort(){
    int l=0,r=0;
    for (int i=1;i<=n;i++)
      if(in[i]==1) q[++r]=i;
    while(l<r) {
        int now=q[++l];
        for (int i=ls[now];i;i=a[i].next){
            int y=a[i].to;
            if(in[y]>1){ //入度>=2的点即为环上的点
                in[y]--;
                if(in[y]==1) q[++r]=y;
            }
        }
    }
}
```

也可以用dfs去找

```
void get_loop(int u) {
    dfn[u] = ++ idx;
    for (int i = 0; i < G[u].size(); i ++) {
        int v = G[u][i];
        if(v == fa[u]) continue ;
        if(dfn[v]) {
            if(dfn[v] < dfn[u]) continue ;
            loop[++ cnt] = v;
            for ( ; v != u; v = fa[v])
                loop[++ cnt] = fa[v];
        } else fa[v] = u, get_loop(v);
    }
}
```

对于有向图

采用dfs一直往深处搜即可，码量小

```
void check_c(int x)
{
    v[x]=true;
    if(v[d[x]]) mark=x; //mark即为
    else check_c(father[x]);
    return;
}
```

习题

[ZJOI2008]骑士(基环树上DP)

N个骑士组个队，每个骑士有战力，每个骑士恨一人，相互憎恨不同在队里，怎样安排战斗力最大？输出最大总战斗力即可。

对每块基环树dp求出答案之和。基环树上的二选一删边较大值就是基环树上的答案。

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
using namespace std;
const int maxn=1e6+7;

//int read(){   int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

struct Edge{
    int nxt,to;
}e[maxn];

int n,deg[maxn],vis[maxn],mrky,mrke;
int f[maxn][2],ans,tmp;
int a[maxn],v[maxn];
int head[maxn],cnt=0;
```

```
inline void addedge(int u,int v){
    e[++cnt]=(Edge){head[u],v};
    head[u]=cnt;
}

void dp(int x){
    vis[x]=1;
    f[x][1]=a[x];f[x][0]=0;
    for(int i=head[x];i;i=e[i].nxt){
        int y=e[i].to;
        if(y==mrky){
            f[y][1]=-inf;
            continue;
        }
        dp(y);
        f[x][0]+=max(f[y][0],f[y][1]);
        f[x][1]+=f[y][0];
    }
}

void find_loop(int x){
    vis[x]=1;
    if(vis[v[x]]) mrky=x; //讨厌的人已经被选了
    else find_loop(v[x]);
    return;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>a[i]>>v[i];
        addedge(v[i],i);
    }
    for(int i=1;i<=n;i++){
        if(vis[i]) continue;
        find_loop(i);
        dp(mrky);
        tmp=max(f[mrky][0],f[mrky][1]);
        mrky=v[mrky];
        dp(mrky);
        ans+=max(tmp,max(f[mrky][0],f[mrky][1]));
    }
    cout<<ans<<"\n";
    return 0;
}
```

[IOI2008] Island(基环树直径)

给出一个基环树森林（每一个点有一条无向边），让你求出所有基环树的直径（即一条不经过重复点的基环树上的最长路径）之和。

对每一块基环树求树的直径相加即可。

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

```cpp
#define int long long
#define inf 0x7f7f7f7f
using namespace std;
const int maxn=1e6+10;

struct Edge{
    int to,w,nxt;
}e[maxn<<1];

int head[maxn],cnt=1;
inline void addedge(int u,int v,int w){ //加入边
    e[cnt]={v,w,head[u]};
    head[u]=cnt++;
}

int n,v,w,ans;
bool vis[maxn];
int fa[maxn],lp[maxn],size; //loop存环，size最终为环的大小
int dfn[maxn],idx;

void get_loop(int u){
    dfn[u]=++idx;   //时间戳
    for(int i=head[u];~i;i=e[i].nxt){
        int v=e[i].to;
        if(v==fa[u]) continue;
        if(dfn[v]){
            if(dfn[v]<dfn[u]) continue; //注意：第一次遇到环时先不存
            lp[++size]=v;               //在dfs的回归过程中再次遇到该环时再记录
            for(;v!=u;v=fa[v]) lp[++size]=fa[v];    //记录环
        }
        else fa[v]=u,get_loop(v);   //继续dfs子节点
    }
}

int d[maxn],ansd;   //d[i]表示从i出发走向以 i 为根的子树，能到达的最远距离

void dp(int u){ //dp求树的直径
    vis[u]=true;
    for(int i=head[u];~i;i=e[i].nxt){
        int v=e[i].to,w=e[i].w;
        if(vis[v]) continue;
        dp(v);
        ansd=max(ansd,d[u]+d[v]+w); //用经过点u的最长链更新ansd
        //最长链即各个 d[v]+w 的最大值和次大值的和，
        //最后一次更新时的 d[u] 一定是次大值，d[v]+w 为最大值
        d[u]=max(d[u],d[v]+w);  //d[u] 应为所有 d[v]+w 中的最大值
    }
}

int s[maxn<<1]; //s为前缀和数组
int q[maxn<<1],l,r; //单调队列

inline int solve(int p){
    idx=size=0;
    get_loop(p);
    int len1=0,len2=0;
    lp[0]=lp[size];
    for(int i=1;i<=size;i++) vis[lp[i]]=true;
```

```
    for(int i=1;i<=size;i++){
        ansd=0; dp(lp[i]);
        len1=max(len1,ansd);
    }//计算出情况一(普通的树)的答案 len1
    if(size==2){    //单独处理由两个点构成的环的情况
        for(int i=head[lp[1]];~i;i=e[i].nxt){
            if(e[i].to==lp[2])
                len2=max(len2,d[lp[1]]+d[lp[2]]+e[i].w);
        }
        return max(len1,len2);
    }
    for(int i=1;i<=size;i++){
        int lpw;
        for(int j=head[lp[i]];~j;j=e[j].nxt)
            if(e[j].to==lp[i-1]){
                lpw=e[j].w; break;
            }//找到边 (lp[i-1],lp[i])
        s[i]=s[i-1]+lpw;    //计算前缀和
    }
    for(int i=1;i<size;i++) s[size+i]=s[size]+s[i]; //复制一倍
    l=r=1; q[1]=0;
    for(int i=1;i<(size<<1);i++){   //单调队列计算情况二的答案 len2
        while(l<=r&&q[l]<=i-size) l++;
        //判断队头的决策是否超出size的范围，超出则出队
        len2=max(len2,d[lp[q[l]%size]]+d[lp[i%size]]+s[i]-s[q[l]]);
        //此时队头即为使 d[lp[i%size]-s[i] 最大的 i，并更新答案
        while(l<=r&&s[q[r]]-d[lp[q[r]%size]]>=s[i]-d[lp[i%size]]) r--;
        //若队尾的值没有当前的 i 更优，出队
        q[++r]=i;   //将i入队
    }
    return max(len1,len2);  //最终答案为两种情况中较大的
}
signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n;
    memset(head,-1,sizeof(head));
    for(int i=1;i<=n;i++){
        cin>>v>>w;
        addedge(i,v,w);
        addedge(v,i,w);
    }
    for(int i=1;i<=n;i++)
        if(!vis[i]) ans+=solve(i);  //计算每个连通块（基环树）的直径并累加
    cout<<ans<<"\n";
    return 0;
}
```

参考资料

https://www.luogu.com.cn/blog/user52918/qian-tan-ji-huan-shu

https://blog.csdn.net/Binary_Heap/article/details/82080267

https://www.luogu.com.cn/blog/user52918/qian-tan-ji-huan-shu