

数据结构优化建图

算法例题

例题一

CF786B

题意

n 个点的有向图。给定 3 中连边方式，分别为：

1. u 向 v 连一条边权为 w 的边
2. u 向 $[l, r]$ 每个节点连一条边权 w 的边
3. $[l, r]$ 向 u 每个节点连一条边权 w 的边

给定源点 s 询问单点源最短路。

题解

考虑建两棵线段树，每棵线段树均维护区间 $[1, n]$

第一棵线段树每个父结点向它的子节点连一条权值为 0 的边，这样 u 向线段树中的 v 连边等价于 u 向 v 的子树连边。

第二棵线段树每个子结点向它的父结点连一条权值为 0 的边，这样线段树中的 v 向 u 连边等价于 v 的子树向 u 连边。

考虑这两棵线段树共享叶子结点，同时用每个叶子结点代表原图中的 $[1, n]$ 结点，于是操作 $2, 3$ 转化为 $O(\log n)$ 的连边操作。

最后跑最短路算法，时间复杂度 $O(m \log^2 n)$

```
const int MAXN=1e5+5;
const LL Inf=1e18;
struct Edge{
    int to,w,next;
}edge[MAXN*30];
int head[MAXN<<3],edge_cnt;
void Insert(int u,int v,int w){
    edge[++edge_cnt]=Edge{v,w,head[u]};
    head[u]=edge_cnt;
}
template <typename T>
struct dijkstra{
    T dis[MAXN<<3];
```

```
bool vis[MAXN<<3];
priority_queue<pair<T,int>,vector<pair<T,int> >,greater<pair<T,int> >
>q;
void solve(int src,int n){
    mem(vis,0);
    _rep(i,1,n)
        dis[i]=Inf;
    dis[src]=0;
    q.push(make_pair(dis[src],src));
    while(!q.empty()){
        pair<T,int> temp=q.top();q.pop();
        int u=temp.second;
        if(vis[u])
            continue;
        vis[u]=true;
        for(int i=head[u];i;i=edge[i].next){
            int v=edge[i].to;
            if(dis[v]>edge[i].w+dis[u]){
                dis[v]=edge[i].w+dis[u];
                q.push(make_pair(dis[v],v));
            }
        }
    }
};
dijkstra<LL> solver;
int lef[MAXN<<2],rig[MAXN<<2],Tree1[MAXN<<2],Tree2[MAXN<<2],node_cnt;
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R;
    int M=L+R>>1;
    if(L==R){
        Tree1[k]=Tree2[k]=M;
        return;
    }
    Tree1[k]=++node_cnt,Tree2[k]=++node_cnt;
    build(k<<1,L,M);
    build(k<<1|1,M+1,R);
    Insert(Tree1[k],Tree1[k<<1],0);Insert(Tree1[k],Tree1[k<<1|1],0);
    Insert(Tree2[k<<1],Tree2[k],0);Insert(Tree2[k<<1|1],Tree2[k],0);
}
void AddEdge1(int k,int L,int R,int v,int w){
    if(L<=lef[k]&&rig[k]<=R)
        return Insert(v,Tree1[k],w);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=L)
        AddEdge1(k<<1,L,R,v,w);
    if(mid<R)
        AddEdge1(k<<1|1,L,R,v,w);
}
void AddEdge2(int k,int L,int R,int v,int w){
```

```

    if(L<=lef[k]&&rig[k]<=R)
    return Insert(Tree2[k],v,w);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=L)
    AddEdge2(k<<1,L,R,v,w);
    if(mid<R)
    AddEdge2(k<<1|1,L,R,v,w);
}
void Init(int n){
    node_cnt=n;
    build(1,1,n);
}
int main()
{
    int n=read_int(),m=read_int(),s=read_int();
    Init(n);
    while(m--){
        int type=read_int();
        if(type==1){
            int u=read_int(),v=read_int(),w=read_int();
            Insert(u,v,w);
        }
        else{
            int v=read_int(),l=read_int(),r=read_int(),w=read_int();
            if(type==2)
                AddEdge1(1,l,r,v,w);
            else
                AddEdge2(1,l,r,v,w);
        }
    }
    solver.solve(s,node_cnt);
    _rep(i,1,n)
    space(solver.dis[i]==Inf?-1:solver.dis[i]);
    return 0;
}

```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001-%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84%E4%BC%98%E5%8C%96%E5%BB%BA%E5%9B%BE&rev=1613008686

Last update: 2021/02/11 09:58