Prim算法优化

优先队列优化

现在时间复杂在寻最近距离现在这个MST最近的点，容易想到使用优先队列进行进行维护，每一次松弛都插入优先队列里面，但是这样可能导致一个点多次入队，最坏情况每一次遍历；临接点都入队了，那么队中最多 $\mathcal{O}(|E|)$ 个元素，优先队列查询 $\mathcal{O}(1)$ 修改 $\mathcal{O}(\log|E|)$ ，总体时间复杂度 $\mathcal{O}(|E|\log|E|+|V|)$ 。

手写堆+堆内修改优化

现在时间复杂度并不理想，考虑再次进行优化，主要是堆内元素有 $\mathcal{O}(|E|)$ 个所以时间复杂度高。考虑一个点已经在堆里面的时候就直接堆内修改，这样保证队内最多 $\mathcal{O}(|V|)$ 个元素，实际上还是可能需要插入 $\mathcal{O}(|E|)$ 次但是堆内最多 $\mathcal{O}(|V|)$ 个元素，由于对所以时间复杂度 $\mathcal{O}(|E|\log|V|+|V|)$

参考实现代码

```cpp
#include<cstdio>
#include<queue>
// #define ONLINE_JUDGE
#define INPUT_DATA_TYPE long long
#define OUTPUT_DATA_TYPE long long
INPUT_DATA_TYPE read(){register INPUT_DATA_TYPE x=0;register char
f=0,c=getchar();while(c<'0'||'9'<c)f=(c=='-'),c=getchar();while('0'<=c&&c<='9')x=
(x<<3)+(x<<1)+(c&15),c=getchar();return f?-x:x;}void print(OUTPUT_DATA_TYPE x)
{register char s[20];register int i=0;if(x<0){x=-x;putchar('-');}if(x==0)
{putchar('0');return;}while(x){s[i++]=x%10;x/=10;}while(i){putchar(s[--
i]+'0');}return;}

struct EDGE{//链式前向星存图
    int to,next;
    long long w;
    EDGE(){
        next=-1;
        return;
    }
}e[1000010];

int tot,head[1010];//链式前向星存图
char book[1010];//标记点的颜色 char压位

struct HEAP_EDGE{//放在堆里面的点
    int v;
    long long w;
    bool operator < (const HEAP_EDGE another) const{//重载<运算
        return w<another.w;
    }
};

void addEdge(int u,int v,long long w){//链式前向星添加边
    e[tot].to=v;
    e[tot].w=w;
    e[tot].next=head[u];//前插加边
    head[u]=tot;//前插加边
    ++tot;
    return;
}

#define HEAP_DATA_TYPE HEAP_EDGE
#define HEAP_COMPARE_TYPE <
const int HEAP_SIZE=1010;

struct HEAP{//手写堆+堆内修改
    HEAP_DATA_TYPE heap[HEAP_SIZE];
    int book[HEAP_SIZE];//记录元素位置，这样可以直接堆内修改
```

```
    int tail;

    HEAP(){//构造函数，初始化
        tail=0;
        return;
    }

    void build(HEAP_DATA_TYPE *data,int n){//On建堆
        register int i;
        tail=n;
        for(i=1;i<=n;++i) heap[i]=data[i];
        for(i=tail>>1;i>=1;--i){
            down(i);
        }
        return;
    }

    void swap(int pos1,int pos2){//交换元素位置
        int temp=book[heap[pos1].v];//交换记录的元素位置
        book[heap[pos1].v]=book[heap[pos2].v];
        book[heap[pos2].v]=temp;
        HEAP_DATA_TYPE temp2=heap[pos1];
        heap[pos1]=heap[pos2];
        heap[pos2]=temp2;
        return;
    }

    bool empty(){
        return size()==0;
    }

    int size(){
        return tail;
    }

    HEAP_DATA_TYPE top(){
        return heap[1];
    }

    void push(HEAP_DATA_TYPE in){//插入元素
        if(book[in.v]){//要插入的元素已经存在
            if(in HEAP_COMPARE_TYPE heap[book[in.v]])//并且插入的元素更优
                insert(in,book[in.v]);
        }else{//不存在放在堆尾
            book[in.v]=++tail;
            insert(in,tail);
        }
        return;
    }
```

```
    void pop(){//弹出元素
        book[heap[tail].v]=0;//清理元素位置记录
        heap[1]=heap[tail--];
        down(1);
        return;
    }

    int down(register int pos){//沉降
        register int next;
        while((pos<<1)<=tail){
            next=pos<<1;
            if(((next|1)<=tail)&&(heap[next|1] HEAP_COMPARE_TYPE heap[next])) next|=1;
            if(heap[next] HEAP_COMPARE_TYPE heap[pos]){
                swap(next,pos);
                pos=next;
            }else break;
        }
        return pos;
    }

    void up(register int pos){//上浮
        while((pos>1)&&(heap[pos] HEAP_COMPARE_TYPE heap[pos>>1])){
            swap(pos,pos>>1);
            pos>>=1;
        }
        return;
    }

    void insert(HEAP_DATA_TYPE data,register int pos){//插入到pos的位置
        heap[pos]=data;
        up(down(pos));
        return;
    }
}h;

int main(){
    #ifndef ONLINE_JUDGE
    freopen("express.in", "r", stdin);
    freopen("express.out", "w", stdout);
    #endif

    register int i,u,v;
    register long long ans=0;
    HEAP_EDGE now;
    long long w;
    int n=read();
    int m=read();
    for(i=0;i<=n;++i) head[i]=-1;
```

```
    for(i=0;i<m;++i){
        u=read();
        v=read();
        w=read();
        addEdge(u,v,w);//无向图双向边
        addEdge(v,u,w);
    }

    h.push((HEAP_EDGE){1,0});//从点1开始构建MST

    while(!h.empty()){//直到堆空
        now=h.top();
        h.pop();
        if(book[now.v]) continue;//已经处理过了
        book[now.v]=1;//标记
        ans+=now.w;
        for(i=head[now.v];~i;i=e[i].next)//松弛
            if(!book[e[i].to])
                h.push((HEAP_EDGE){e[i].to,e[i].w});
    }

    print(ans);

    #ifndef ONLINE_JUDGE
    fclose(stdin);
    fclose(stdout);
    #endif
    return 0;
}
```