Happy Triangle(动态开点线段树)

维护一个可重集合 S，要求支持一下操作：

- 插入一个数 x
- 删除一个数 x
- 对于给定的数 x，查询是否存在 $a,b \in S$，使得 x,a,b 作为边长能构

思路：动态开点线段树+map维护集合。

对于询问$1,2$的插入和删除操作用map维护起来就行了。

对于询问3，我们只需要找到a,b不妨设$a \leq b$,使得a,b,x组成三角形。

显然若a,b,x能组成三角形，则$b',b,x,(b' \in (a,b])$也组成三角形。

因为$x \in (b-a,a+b),b-b' < b-a,b+b' > a+b$，使得区间更大，更能满足情况，综上即取$b$的前驱即可。

因此我们考虑用线段树维护每个数与前驱差值，因为数据范围有$1e9$，所以考虑离散化或者动态开点。

对每次询问查询第一个大于等于$\frac{x}{2}+1$的数，然后判断一下最小差值是否小于x，查询第一个大于等于$\frac{x}{2}+1$的数保证了两数之和大于x，即上界，然后后面的判断保证了下界。

时间复杂度：$O(nlogn)$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1e6+100,M=N*40,inf=1e9,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
#define pb push_back
int m[M],ls[M],rs[M],cnt,rt;
map<int,int>mp;
void upd(int &id,int l,int r,int x,int val){
    if(!id) id=++cnt,m[id]=val;
    if(l==r){m[id]=val;return;}
    int mid=l+r>>1;
    if(x<=mid) upd(ls[id],l,mid,x,val);
    else upd(rs[id],mid+1,r,x,val);
    int ans=2e9;
    if(ls[id]&&m[ls[id]]<ans) ans=m[ls[id]];
    if(rs[id]&&m[rs[id]]<ans) ans=m[rs[id]];
    m[id]=ans;
}
void add(int x){
    mp[x]++;
```

```
    if(mp[x]==1){
        auto it=mp.lower_bound(x);
        ++it;
        if(it!=mp.end()&&it->se==1)
            upd(rt,0,inf,it->first,it->first-x);
        --it;
        int pre=-2e9;
        if(it!=mp.begin()) pre=(--it)->first;
        upd(rt,0,inf,x,x-pre);
    }
    else if(mp[x]==2) upd(rt,0,inf,x,0);
}
void del(int x){
    int pre=-1e9;
    auto it=mp.lower_bound(x);
    mp[x]--;
    if(it!=mp.begin()) pre=(--it)->fi,++it;
    if(!mp[x]){
        if((++it)!=mp.end()&&it->se==1){
            upd(rt,0,inf,it->fi,it->fi-pre);
        }
        upd(rt,0,inf,x,2e9);
        mp.erase(x);
    }
    else if(mp[x]==1) upd(rt,0,inf,x,x-pre);
}
int ask(int x){
    auto it=mp.lower_bound(x/2+1);
    if(it==mp.end()) return 2e9;
    if(it->se>1) return it->first;
    if(it!=mp.begin()){ //这里是begin()
        auto pre=it;--pre;
        if(pre->fi+it->fi>x) return it->fi;
    }
    if((++it)!=mp.end()) return it->fi;
    return 2e9;
}
int ask_min(int id,int l,int r,int L,int R){
    if(!id||l>r) return 2e9;
    if(L<=l&&R>=r) return m[id];
    int ans=2e9;
    int mid=l+r>>1;
    if(L<=mid) ans=min(ans,ask_min(ls[id],l,mid,L,R));
    if(R>mid) ans=min(ans,ask_min(rs[id],mid+1,r,L,R));
    return ans;
}
int main(){
    int q,op,x;
    scanf("%d", &q);
      while(q--){
            scanf("%d%d", &op, &x);
            if(op == 1) add(x);
            if(op == 2) del(x);
            if(op == 3) {
                  if(ask_min(1,0,inf,ask(x),1e9) < x) puts("Yes");
                  else puts("No");
```

```
            }
        }
    return 0;
}
```