

```

222 Persistent Segment Tree:
223 #define left l , (l+r)>>1
224 #define right ((l+r)>>1)+1 , r
225
226 struct PersistentSegTree{
227     struct Vertex {
228         Vertex *l, *r;
229         int sum;
230
231         Vertex(int val) : l(nullptr), r(nullptr), sum(val) {}
232         Vertex(Vertex *l, Vertex *r) : l(l), r(r), sum(0) {
233             if (l) sum += l->sum;
234             if (r) sum += r->sum;
235         }
236         //to change marge
237     };
238
239     int n;
240     vector<int> &a;
241     vector<Vertex*> roots;
242
243     PersistentSegTree(int n , vector<int> &a) : a(a) {
244         this->n = n;
245         roots.push_back(build(0 , n-1));
246     }
247
248     Vertex* build(int l , int r){
249         if(l==r) return new Vertex(a[l]);
250         return new Vertex(build(left) , build(right));
251     }
252
253     //point update
254     void update(int i , int val , int rootIndex){
255         roots.push_back(
256             update(i , val , roots[rootIndex] , 0 , n-1)
257         );
258     }
259
260     Vertex* update(int i , int val , Vertex* p , int l , int r){
261         if(l==r) return new Vertex(val); //to change
262         if(i ≤ (l+r)>>1)
263             return new Vertex(update(i , val , p->l , left) , p->r);
264         else
265             return new Vertex(p->l , update(i , val , p->r , right));
266     }
267
268     //use this function
269     int query(int i , int j , int rootIndex){
270         return query(i , j , roots[rootIndex] , 0 , n-1);
271     }
272
273     int query (int i , int j , Vertex* p , int l , int r){
274         if(j<l || r<i) return 0; // to change
275         if(i≤l && r≤j) return p->sum;
276         return (query(i , j , p->l , left) + query(i , j , p->r , right)); //to
change
277     }
278 };

```