知识精炼（一）

主讲人：邓哲也

POJ 2107 K-th Number

问题：有一个长度为 n 的序列，a[1], a[2], …, a[n]。
每次询问区间[i, j]中的第 k 小的数。

n <= 100000, 询问次数 <= 5000

样例：

7 3	输出：
1 5 2 6 3 7 4	5
2 5 3	6
4 4 1	3
1 7 3	

POJ 2107 K-th Number

区间第 k 小的数。

一个很自然的想法，我们二分答案 x，查询一个区间内有几个数比 x 小。

对值的范围建一颗线段树，把区间中的数全都插入线段树，查询 [1, x-1] 的和即可。

POJ 2107 K-th Number

对值的范围建一颗线段树，把区间中的数全都插入线段树。

此时不需要在外层二分，直接在树上二分即可。

if sum[ls] < k: goto rs, k -= sum[ls]

else: goto ls

最后走到的叶节点就是答案。

POJ 2107 K-th Number

可是询问的区间是不同的，每次都有一个不同的[l, r]。

如何建出一颗线段树包含 a[l]..a[r] 这些值呢？

联系可持久化线段树。

T[0] 是一颗空的线段树。

T[1] 是在T[0]中插入a[1]

T[2] 是在T[1]中插入a[2]

…

T[i] 是在T[i-1]中插入a[i]

POJ 2107 K-th Number

如此以来，对于区间[l, r]

T[r] 减去 T[l-1] 的值，得到的线段树 S，就是包含了所有

a[l]..a[r] 的线段树。

具体实现时，只要维护两个指针 x 和 y

x 指向 T[l-1] 的根节点

y 指向 T[r] 的根节点

每次用 sum[y]-sum[x] 就等价于在线段树 S 上对应节点的值。

采用树上二分的算法找第k大。

时间和空间复杂度均为 O(n log n)

代码实现

```
void modify(int p, int v, int l, int r, int &x){
    x = ++ tot;
    lc[x] = lc[p];
    rc[x] = rc[p];
    sum[x] = sum[p] + 1;
    if (l == r) return;
    int mid = (l + r) >> 1;
    if (v <= mid) modify(lc[p], v, l, mid, lc[x]);
    else modify(rc[p], v, mid + 1, r, rc[x]);
}
```

代码实现

```
int query(int k, int l, int r, int x, int y){
    if (l == r) return l;
    int s = sum[lc[y]] - sum[lc[x]], mid = (l + r) >> 1;
    if (s < k) return query(k - s, mid + 1, r, rc[x], rc[y]);
    else return query(k, l, mid, lc[x], lc[y]);
}
```

代码实现

```
void solve(){
    tot = 0;
    scanf("%d%d", &n, &m);
    for(int i = 1;i <= n;i ++) scanf("%d", &a[i]), bin[i] = a[i], T[i] = 0;
    sort(bin + 1, bin + n + 1);
    int cnt = unique(bin + 1, bin + n + 1) - bin - 1;
    T[0] = 0;
    for (int i = 1;i <= n;i ++){
        a[i] = lower_bound(bin + 1, bin + cnt + 1, a[i]) - bin;
        modify(T[i - 1], a[i], 1, cnt, T[i]);
    }
    int x, y, k;
    while(m --){
        scanf("%d%d%d", &x, &y, &k);
        printf("%d\n", bin[query(k, 1, cnt, T[x - 1], T[y])]);
    }
}
```

下节课再见