信息学

RMQ问题与ST表

西南大学附属中学校

信息奥赛教练组

RMQ问题

在学习中，常常有这样一种问题

给定长度为n的数列，m次询问，每次给定L, R ≤ n，查询区间[L, R]内的最值。
n,m都很大

Range **M**aximun/Minimum **Q**uery，区间最值问题

RMQ问题的解决办法

- 朴素算法：搜索
- 线段树
- ST表
- RMQ标准算法

	初始化时间复杂度	总查询时间复杂度
朴素算法：搜索	O(n)	O(nm)
线段树	O(n)	O(mlogn)
ST表	O(nlogn)	O(m)
RMQ标准算法	O(n)	O(m)

ST表原理

ST算法原理：一个大区间若能被两个小区间覆盖，那么大区间的最值等于两个小区间的最值。例如下图中，

小区间重叠不影响大区间的答案
这个性质决定是否能用ST表解决

从以上原理得到ST算法的基本思路，包括两个步骤：

（1）把整个数列分为很多小区间，并提前计算出每个小区间的最值；

（2）对任意一个区间最值查询，找到覆盖它的两个小区间，用两个小区间的最值算出答案。

问题：如何分区间最好?

倍增 $N = a_0 2^0 + a_1 2^1 + a_2 2^2 + a_3 2^3 + a_4 2^4 + \dots$

倍增法的特点是需要提前计算出第1、2、4、…、2^k 个跳板，这要求数据是静态不变的，不是动态变化的。如果数据发生了变化，所有跳板要重新计算，跳板就失去了意义。

ST表

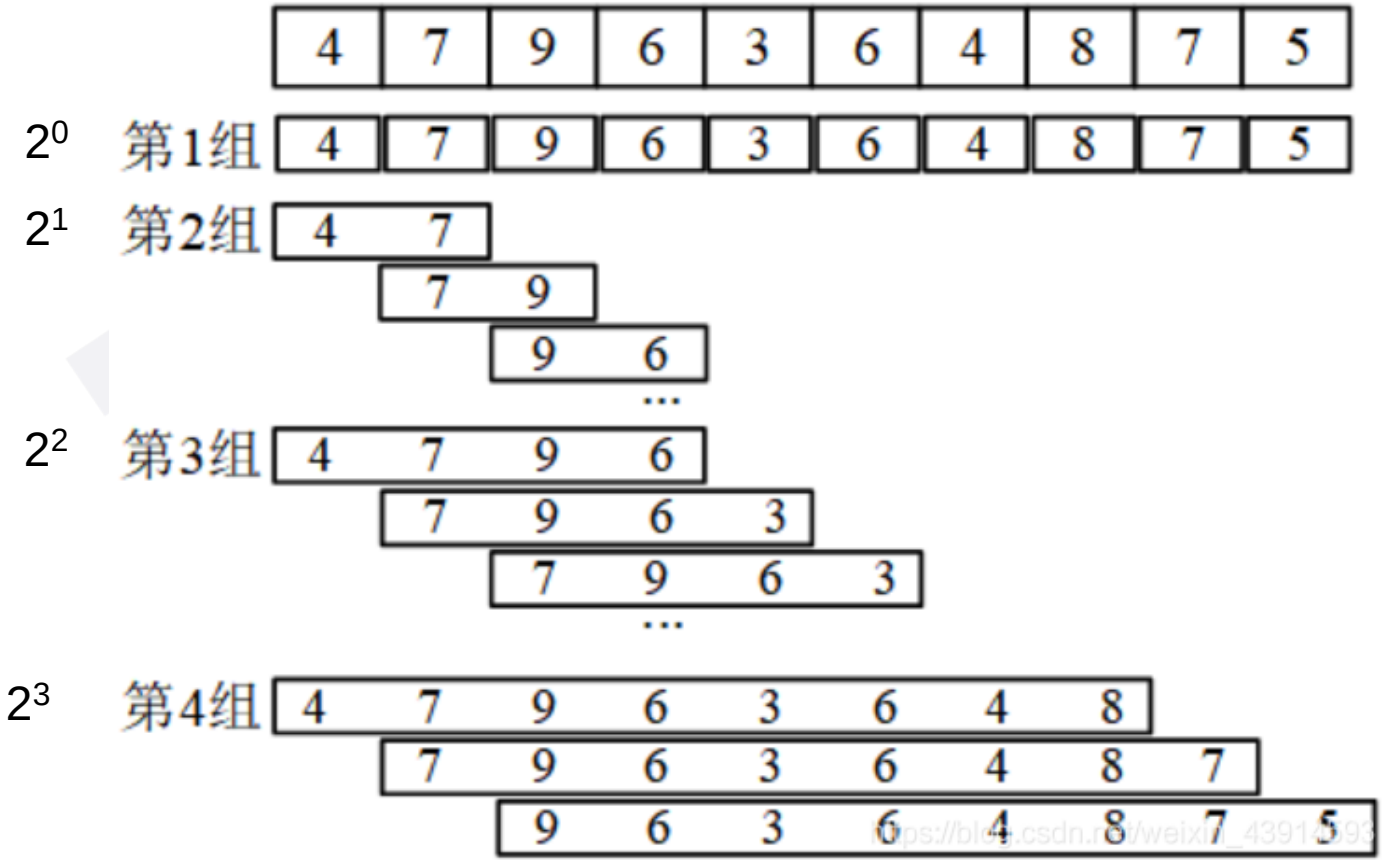
一般RMQ的Sparse Table（ST）算法是基于倍增思想设计的$O(N\log_2 N)$ 离线处理– O(1)在线查询算法

算法记录从每个元素开始的连续的长度为2^k的区间中元素的最小值，并以在常数时间内解决询问

在线算法：可以输入一个处理一个，不需要知道所有数据(例如：插入排序)
离线算法：需要知道所有数据才能处理（例如：选择排序）

ST表

第一步：倍增分组处理

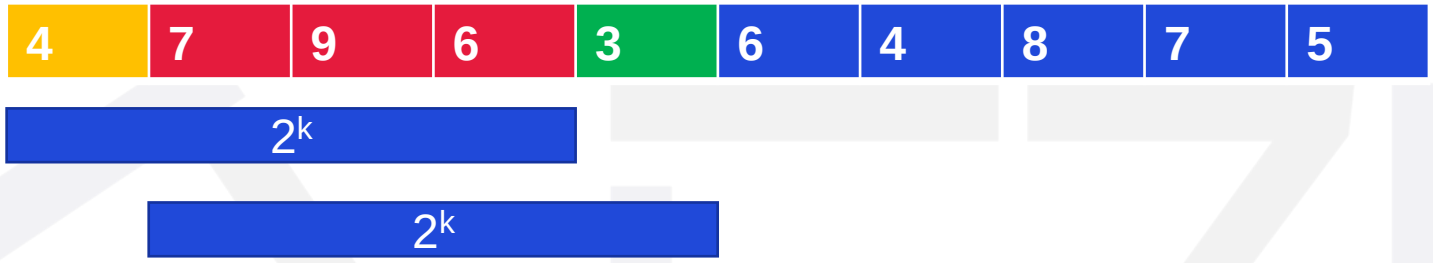


一共可分为logn组
每组n个长度为2^k区间

预处理时间复杂度O(nlogn)

ST表

第二步：求区间[1,5]的最大值



K怎么求?

区间[L, R]的长度是len = R-L+1。两个小区间的长度是x，令x是比len小的最大2的倍数，有2*x ≥ len，这样保证能覆盖。

```
int k=log2(R-L+1); //以2为底的库函数log2()
```

```
LOG2[0] = -1;  //手工算log2，比上面快
for(int i=1;i<=maxn;i++)
    LOG2[i] = LOG2[i>>1]+1;
```

ST表

从左端点覆盖2^k

从右端点覆盖2^k

实质

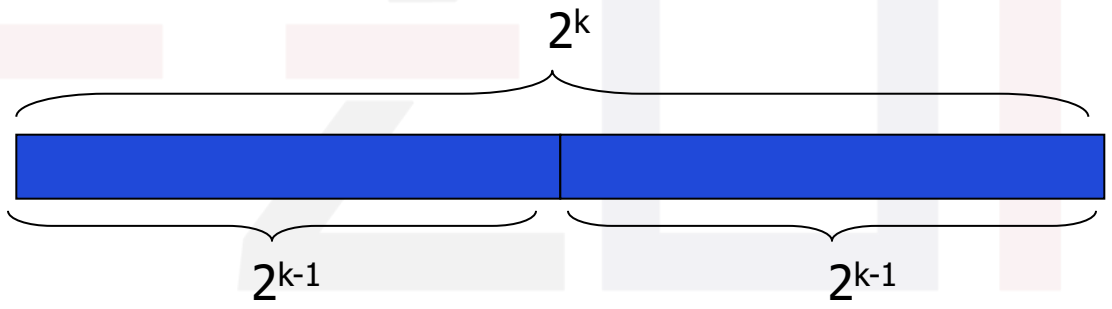
动态规划

状态

`f(i,j)`

表示：区间$[i,i+2^j]$区间中的最小值

`f[i,0] = a[i]`



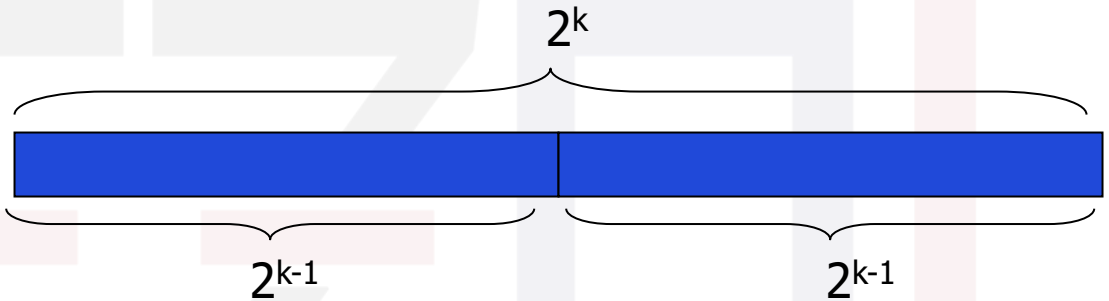
ST表

F[i,j]建立

转移方程f[i,j]=min{f[i,j-1],f[i+2^{j-1},j-1]}

时间复杂度：

O(nlogn)



ST表

F[i,j]使用：

求区间[m,n]的最小值：

1）找到区间[m,n]中a的最小值，找到一个数k使得2^k<n-m+1;

2)区间被分为：[m,m+2^k-1]和[n-2^k+1,n]

3)比较两个区间最小值，即为整个区间最小值;

时间复杂度：

O(1)

例题：Balanced Lineup

题目描述：给定一个包含n个整数的数列，和q个区间询问，询问区间内最大值和最小值的差。

输入：
第一行是2个整数，n和q。
接下来n行，每行一个整数hi。
再后面q行，每行2个整数a、b，表示一个区间询问。

输出：对每个区间询问，返回区间内最大值和最小值的差。

数据范围：$1 \le n \le 5 \times 10^4$，$1 \le q \le 1.8 \times 10^5$，$1 \le a \le b \le n$。

ST表模板题

Balanced Lineup参考代码


```
#include<bits/stdc++.h>
using namespace std;
const int maxn=50005;
int a[maxn],dp_max[maxn][22],dp_min[maxn][21],n,m;
int LOG2[maxn];              //自己计算以2为底的对数，向下取整
void st_init(){  //建立st表
    LOG2[0]=-1;
    for(int i = 1;i<=maxn;i++)  //不用系统的log()函数，自己算
        LOG2[i] = LOG2[i>>1]+1;

    for(int i=1;i<=n;i++){ //初始化区间长度为1时的值
        dp_min[i][0]=a[i];
        dp_max[i][0]=a[i];
    }
    int p=log2(n);     //可倍增区间的最大次方：2^p <= n
     for(int k=1;k<=p;k++)          //倍增计算小区间。先算小区间，再算大区间，逐步递推
        for(int s=1;s+(1<<k)<=n+1;s++){
            dp_max[s][k]=max(dp_max[s][k-1], dp_max[s+(1<<(k-1))][k-1]);
            dp_min[s][k]=min(dp_min[s][k-1], dp_min[s+(1<<(k-1))][k-1]);
   }
}
int st_query(int L,int R){
    //int k=log2(R-L+1);                        //求k
    int k=LOG2[R-L+1];                          //用自己算的LOG2
    int x=max(dp_max[L][k],dp_max[R-(1<<k)+1][k]); //区间最大
    int y=min(dp_min[L][k],dp_min[R-(1<<k)+1][k]); //区间最小
    return x-y;  //返回差值
}
```

```
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)    scanf("%d",&a[i]);
    st_init();
            for(int i=1;i<=m;i++){
                    int L,R; scanf("%d%d",&L,&R);
                    printf("%d\n",st_query(L,R));
            }
            return 0;
}
```

ST表

ST表只能做区间最值问题吗?

只要满足小区间重复覆盖不影响大区间答案这样的性质的问题都可以尝试一下ST表

比如RGQ问题，区间公共公约数问题（例题:**数字对**）。

总结

- ST算法与线段树求解RMQ问题更常用的是线段树，它求RMQ的时间复杂度与ST差不多，但是两者有很大区别：线段树用于动态数组，ST用于静态数组。

- ST算法用O(nlogn)时间来预处理数组，预处理之后每次区间查询是O(1)的。如果数组是动态改变的，改变一次就需要用O(nlogn)预处理一次，导致效率很低。线段树适用于动态维护（添加或删除）的数组，它每次维护数组和每次查询都是O(logn)的。从数组是否能动态维护

- ST是离线算法，线段树是在线算法。ST的优势是编码简单，如果数组是静态的，就用ST。

- ST算法的适用场合从ST算法的原理看，它的核心思想是“大区间被两个小区间覆盖、小区间的重复覆盖不影响结果”，然后用倍增法划分小区间和计算小区间的最值。最大值和最小值符合这种场景，类似的有RGQ问题（Range GCD Query，区间最大公约数问题）：查询给定区间的GCD。

Thanks

For Your Watching