



# ST 表

ST 表 (Sparse Table, 稀疏表) 是一种用于高效解决可重复贡献问题的数据结构，主要用于静态区间查询（如区间最值、区间 GCD 等）。其核心思想是倍增预处理和动态规划，支持  $O(n \log n)$  预处理和  $O(1)$  查询。

## 核心概念

1. 可重复贡献问题：满足  $f(x, x) = x$  且  $f(f(a, b), f(c, d)) = f(a, f(b, f(c, d)))$  的运算（如  $\min$ ,  $\max$ ,  $\gcd$ ）。
  2. 倍增思想：将区间拆分为两个重叠的子区间，通过预处理结果合并得到原区间答案。
- 

## 实现原理

### 预处理 ( $O(n \log n)$ )

1. 状态定义：  
 $st_{i,j}$  表示区间  $[i, i + 2^j - 1]$  的最值（以最小值为例）。
2. 状态转移：

$$st_{i,j} = \min(st_{i,j-1}, st_{i+2^{j-1}, j-1})$$

- 将区间  $[i, i + 2^j - 1]$  拆分为  $[i, i + 2^{j-1} - 1]$  和  $[i + 2^{j-1}, i + 2^j - 1]$  两部分。
3. 初始化：
    - $j = 0$  时， $st_{i,0} = a_i$ （区间长度为 1）。

### 查询 ( $O(1)$ )

查询区间  $[l, r]$  的最值：

1. 计算  $k = \lfloor \log_2(r - l + 1) \rfloor$ 。
2. 结果为：

$$\min(st_{l,k}, st_{r-2^k+1,k})$$

- 两个子区间  $[l, l + 2^k - 1]$  和  $[r - 2^k + 1, r]$  覆盖  $[l, r]$  且可能有重叠（可重复贡献性保证正确性）。
- 

## 核心代码

```
namespace st{
    int stt[maxn][25];
    inline void init(int n){
        for(int i=1;i<=n;i++)stt[i][0] = a[i];
```

```

        for(int j = 1;j <= __lg(n);j++)
            for(int i=1;i+(1<<j)-1<=n;i++)
                stt[i][j] = max(stt[i][j-1],stt[i+(1<<(j-1))][j-1]);
    }
    int find(int l,int r){
        register int s = __lg(r-l+1);
        return max(stt[l][s],stt[r-(1<<s)+1][s]);
    }
}

```

## 例题

P3865 【模板】ST 表 && RMQ 问题

```

#include <bits/stdc++.h>
#define int long long
using namespace std;
const int maxn = 1e5+5;
int n,m,l,r,a[maxn];
namespace st{
    int stt[maxn][25];
    inline void init(int n){
        for(int i=1;i<=n;i++)stt[i][0] = a[i];
        for(int j = 1;j <= __lg(n);j++)
            for(int i=1;i+(1<<j)-1<=n;i++)
                stt[i][j] = max(stt[i][j-1],stt[i+(1<<(j-1))][j-1]);
    }
    int find(int l,int r){
        register int s = __lg(r-l+1);
        return max(stt[l][s],stt[r-(1<<s)+1][s]);
    }
}

signed main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
        scanf("%d",&a[i]);
    st::init(n);
    for (int i = 1; i <= m; i++) {
        scanf("%d%d",&l,&r);
        printf("%d\n",st::find(l,r));
    }
    return 0;
}

```