

14-B2

排序

选取：中位数

中也者，天下之大本也；和也者，天下之达道也

德性是两种恶——过度与不及——的中间。在感情与实践中，恶要么达不到正确，要么超过正确。德性则找到并且选取那个正确。所以虽然从其本质或概念来说德性是适度，从最高善的角度来说，它是一个极端

邓俊辉

deng@tsinghua.edu.cn

归并向量的中位数

❖ 任给有序向量 S_1 和 S_2 , 长度 n_1 和 n_2

如何快速找出 $S = S_1 \cup S_2$ 的中位数?

❖ 蛮力: 经归并得到有序向量 S

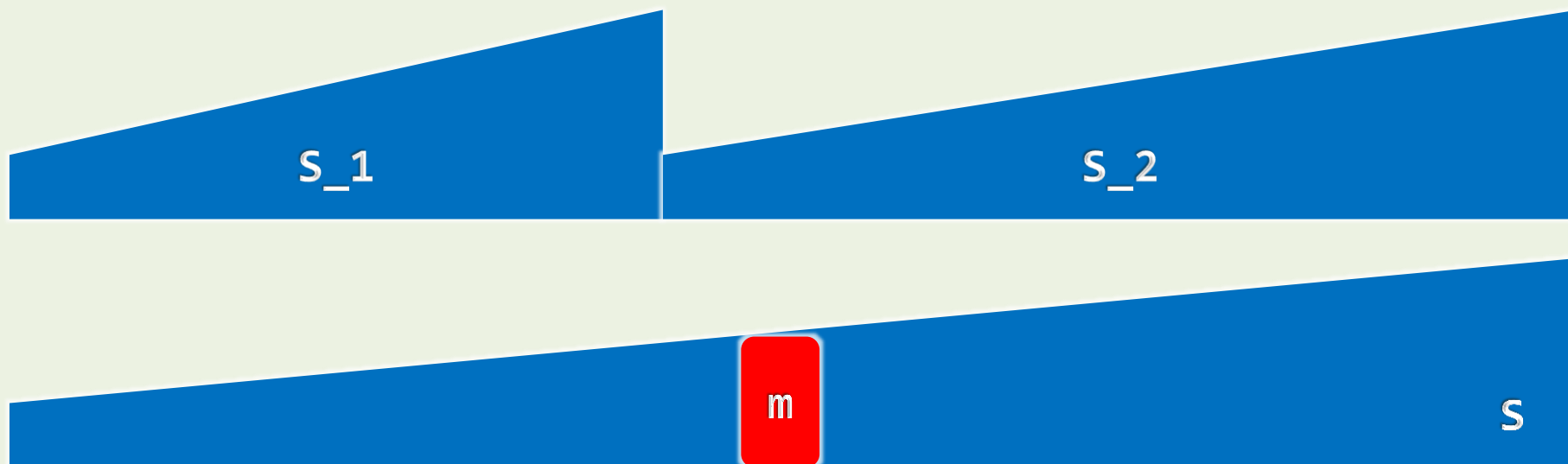
取 $S[(n_1 + n_2)/2]$ 即是

❖ 如此, 共需 $\mathcal{O}(n_1 + n_2)$ 时间

但毕竟未能充分利用 S_1 和 S_2 的有序性

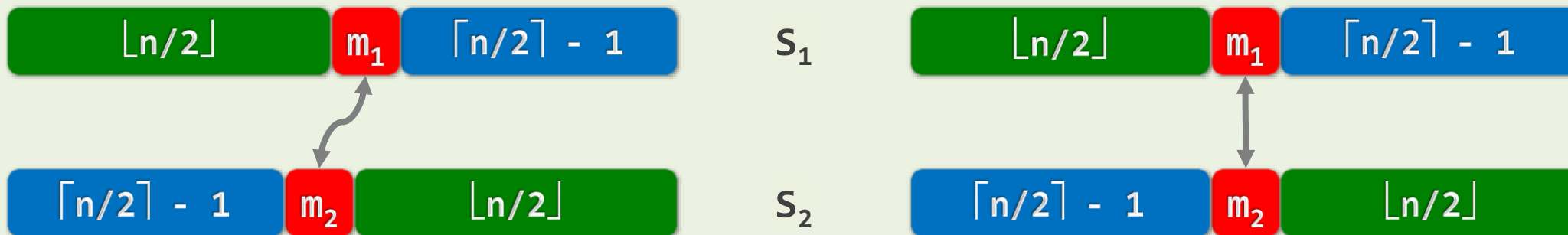
❖ 以下, 先解决 $n_1 = n_2$ 的情况

依然采用减而治之策略...



等长子向量：构思

❖ 考查: $m_1 = S_1[\lfloor n/2 \rfloor]$ 和 $m_2 = S_2[\lceil n/2 \rceil - 1] = S_2[\lfloor (n-1)/2 \rfloor]$



❖ 若 $m_1 = m_2$, 则它们同为 S_1 、 S_2 和 S 的中位数

❖ 若 $m_1 < m_2$, 则 n 无论偶奇, 必有:

$$\text{median}(S_1 \cup S_2) = \text{median}(S_1.\text{suffix}(\lceil n/2 \rceil) \cup S_2.\text{prefix}(\lceil n/2 \rceil))$$

这意味着, 如此减除 (一半规模) 之后, 中位数不变

❖ $m_1 > m_2$ 时同理

等长子向量：实现

```
template <typename T> //尾递归, 可改写为迭代形式
T median( Vector<T> & S1, int lo1, Vector<T> & S2, int lo2, int n ) {
    if ( n < 3 ) return trivialMedian( S1, lo1, n, S2, lo2, n ); //递归基
    int mi1 = lo1 + n/2, mi2 = lo2 + (n - 1)/2; //长度减半
    if ( S1[ mi1 ] < S2[ mi2 ] ) //取S1右半、S2左半
        return median( S1, mi1, S2, lo2, n + lo1 - mi1 );
    else if ( S1[ mi1 ] > S2[ mi2 ] ) //取S1左半、S2右半
        return median( S1, lo1, S2, mi2, n + lo2 - mi2 );
    else
        return S1[ mi1 ];
}
```

任意子向量：实现 (1/2)

```
template <typename T>
```

```
T median ( Vector<T> & S1, int lo1, int n1, Vector<T> & S2, int lo2, int n2 ) {
```

```
    if ( n1 > n2 )
```

```
        return median( S2, lo2, n2, S1, lo1, n1 ); //确保n1 <= n2
```

```
    if ( n2 < 6 )
```

```
        return trivialMedian( S1, lo1, n1, S2, lo2, n2 ); //递归基
```

```
    if ( 2 * n1 < n2 )
```

```
        return median( S1, lo1, n1, S2, lo2 + (n2-n1-1)/2, n1+2-(n2-n1)%2 );
```

任意子向量：实现 (2/2)

```
int mi1 = lo1 + n1/2, mi2a = lo2 + (n1 - 1)/2, mi2b = lo2 + n2 - 1 - n1/2;
```

```
if ( S1[ mi1 ] > S2[ mi2b ] ) //取S1左半、S2右半
```

```
    return median( S1, lo1, n1 / 2 + 1, S2, mi2a, n2 - (n1 - 1) / 2 );
```

```
else if ( S1[ mi1 ] < S2[ mi2a ] ) //取S1右半、S2左半
```

```
    return median( S1, mi1, (n1 + 1) / 2, S2, lo2, n2 - n1 / 2 );
```

```
else //S1保留, S2左右同时缩短
```

```
    return median( S1, lo1, n1, S2, mi2a, n2 - (n1 - 1) / 2 * 2 );
```

```
} //O( log(min(n1,n2)) )——可见，实际上等长版本才是难度最大的
```