

排序

快速排序：迭代、贪心与随机

14-A3

瑕不掩瑜，瑜不掩瑕，忠也

察一叶而知天下秋

邓俊辉

deng@tsinghua.edu.cn

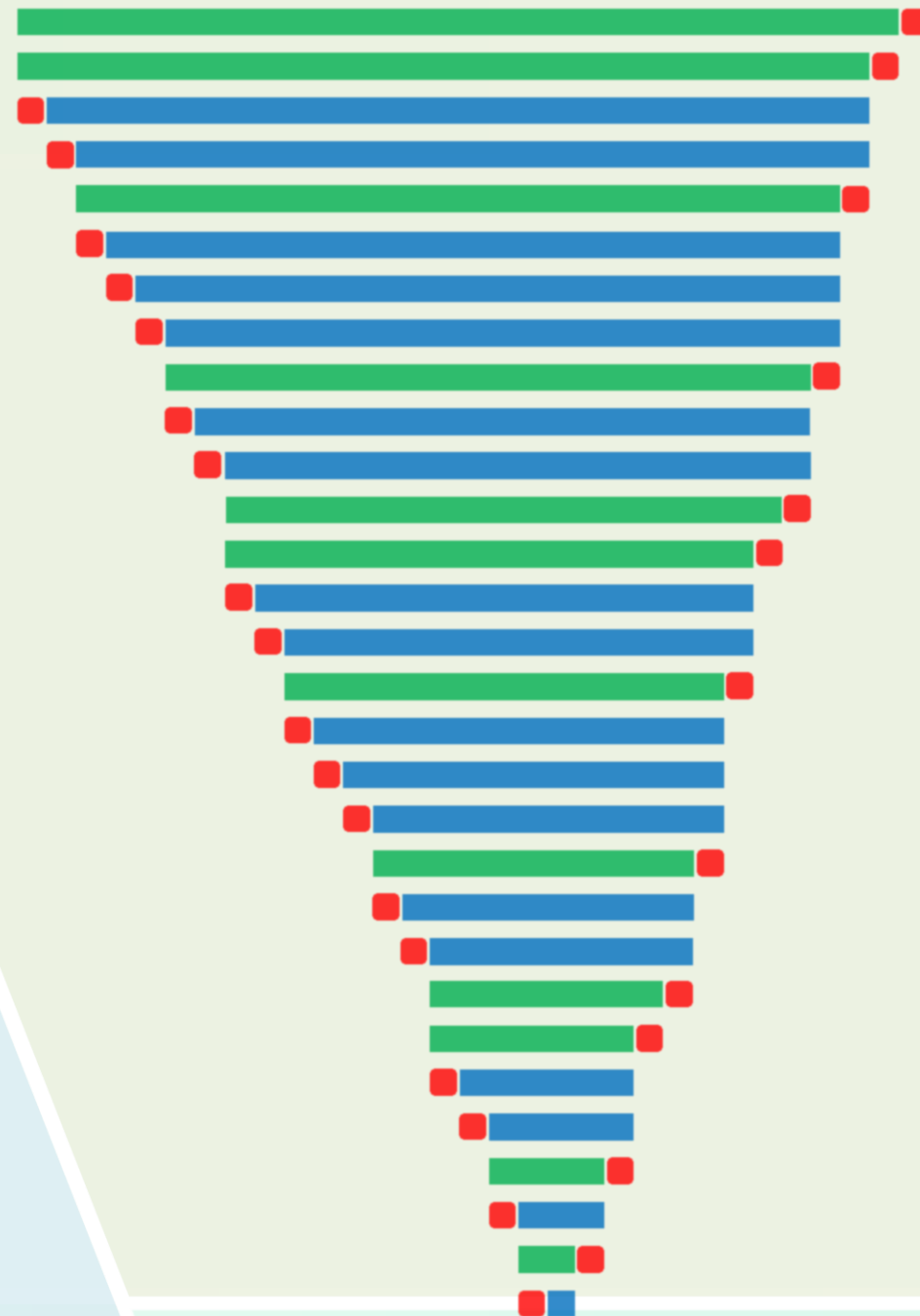
# 空间复杂度 ~ 递归深度

❖ 最好：划分总均衡  $\mathcal{O}(\log n)$

最差：划分皆偏侧  $\mathcal{O}(n)$

平均：均衡不致太少  $\mathcal{O}(\log n)$

❖ 可否避免最坏情况？如何避免？



## 非递归 + 贪心

```
#define Put( K, s, t ) { if ( 1 < (t) - (s) ) { K.push(s); K.push(t); } }
```

```
#define Get( K, s, t ) { t = K.pop(); s = K.pop(); }
```

```
template <typename T> void Vector<T>::quickSort( Rank lo, Rank hi ) {
```

```
    Stack<Rank> Task; Put( Task, lo, hi ); //等效于对递归树的先序遍历
```

```
    while ( ! Task.empty() ) {
```

```
        Get( Task, lo, hi ); Rank mi = partition( lo, hi );
```

```
        if ( mi-lo < hi-mi ) { Put( Task, mi+1, hi ); Put( Task, lo, mi ); }
```

```
        else { Put( Task, lo, mi ); Put( Task, mi+1, hi ); }
```

```
    } //大任务优先入栈（小任务优先出栈执行），可保证递归深度（空间成本）不过 $O(\log n)$ 
```

```
}
```

## 时间性能 + 随机

❖ 最好情况：每次划分都（接近）**平均**，轴点总是（接近）**中央**

$$T(n) = 2 \cdot T\left(\frac{n-1}{2}\right) + \mathcal{O}(n) = \mathcal{O}(n \cdot \log n)$$

到达下界！

❖ 最坏情况：每次划分都**极不均衡**（比如，轴点总是最小/大元素）

$$T(n) = T(n-1) + T(0) + \mathcal{O}(n) = \mathcal{O}(n^2)$$

与起泡排序相当！

❖ 采用**随机选取**（Randomization）、**三者取中**（Sampling）之类的策略

只能**降低**最坏情况的概率，而无法**杜绝**——既如此，为何还称作**快速**排序？