

四牡孔阜，六轡在手
騏驎是中，騶駼是驂
龙盾之合，鋌以鱗輶

曰两美其必合兮，孰信修而慕之？
思九州岛之博大兮，岂惟是其有女？

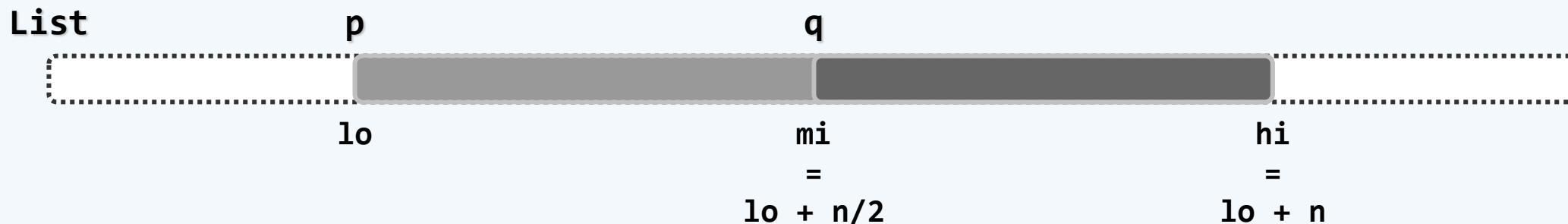
3. 列表

(f) 归并排序

邓俊辉

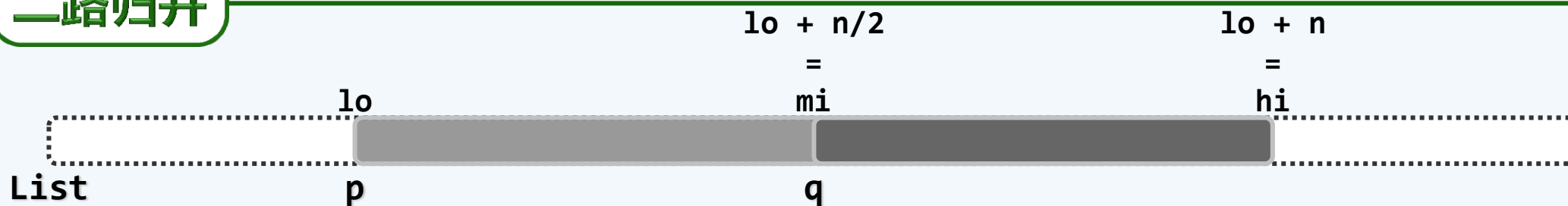
deng@tsinghua.edu.cn

归并排序



```
template <typename T> //valid(p) && rank(p) + n <= size
void List<T>::mergeSort(Posi(T)& p, int n) { //对起始于位置p的n个元素排序
    if (n < 2) return; //待排序范围足够小时直接返回，否则...
    Posi(T) q = p; int m = n >> 1; //以中点为界
    for (int i = 0; i < m; i++) q = q->succ; //均分列表
    mergeSort(p, m); mergeSort(q, n - m); //子序列分别排序
    merge(p, m, *this, q, n - m); //归并
} //若归并可在线性时间内完成，则总体运行时间亦为O(nlogn)
```

二路归并



//当前列表中自p起的n个元素，与列表L中自q起的m个元素归并

```
template <typename T> // ( 归并排序时为同一列表, this == L )
```

```
void List<T>::merge(Posi(T)& p, int n, List<T>& L, Posi(T) q, int m) {
```

```
    while ( 0 < m ) //在q尚未移出区间之前
```

```
        if ( (0 < n) && (p->data <= q->data) ) //若p仍在区间内且v(p) <= v(q)
```

```
            { if ( q == (p = p->succ) ) break; n--; } //则将p直接后移
```

```
        else //若p已超出右界或v(q) < v(p), 则将q插至p之前
```

```
            { insertB( p, L.remove( ( q = q->succ )->pred) ); m--; }
```

```
    } //每经过一次迭代n + m必减少1, 故总体运行时间为O(n + m), 线性正比于节点总数
```