03-H

列表

归并排序

曰两美其必合兮，孰信修而慕之？

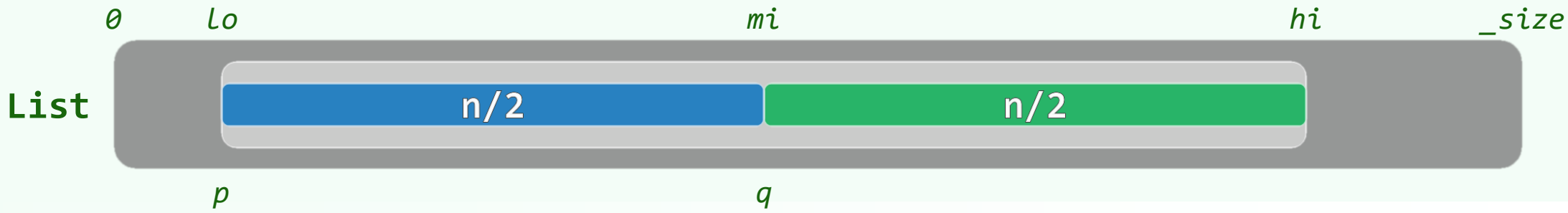
思九州岛之博大兮，岂惟是其有女？

邓俊辉

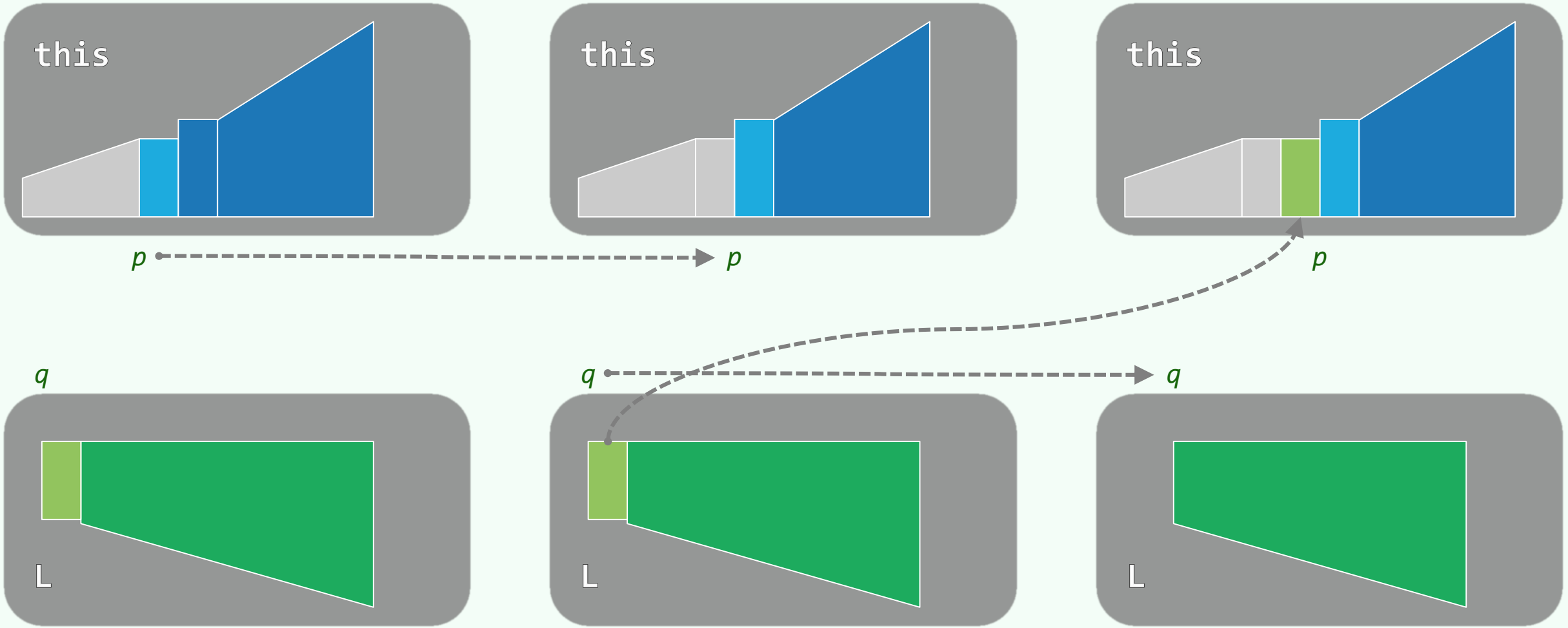
deng@tsinghua.edu.cn

主算法

```
template <typename T> void List<T>::mergeSort( ListNodePosi<T> & p, Rank n ) {
   if ( n < 2 ) return; //待排序范围足够小时直接返回，否则...
   ListNodePosi<T> q = p; Rank m = n >> 1; //以中点为界
   for ( Rank i = 0; i < m; i++ ) q = q->succ; //均分列表：O(m) = O(n)
   mergeSort( p, m ); mergeSort( q, n - m ); //子序列分别排序
   p = 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：算法



二路归并：实现

```
template <typename T> ListNodePosi<T> //this.[p +n) & L.[q +m)：归并排序时，L == this
List<T>::merge( ListNodePosi<T> p, Rank n, List<T>& L, ListNodePosi<T> q, Rank m ) {
   ListNodePosi<T> pp = p->pred; //归并之后p或不再指向首节点，故需先记忆，以便返回前更新
   while ( ( 0 < m ) && ( q != p ) ) //小者优先归入
      if ( ( 0 < n ) && ( p->data <= q->data ) ) { p = p->succ; n--; } //p直接后移
      else { insert( L.remove( ( q = q->succ )->pred ) , p ); m--; } //q转至p之前
   return pp->succ; //更新的首节点
} //运行时间O(n + m)，线性正比于节点总数
```