列表

归并排序

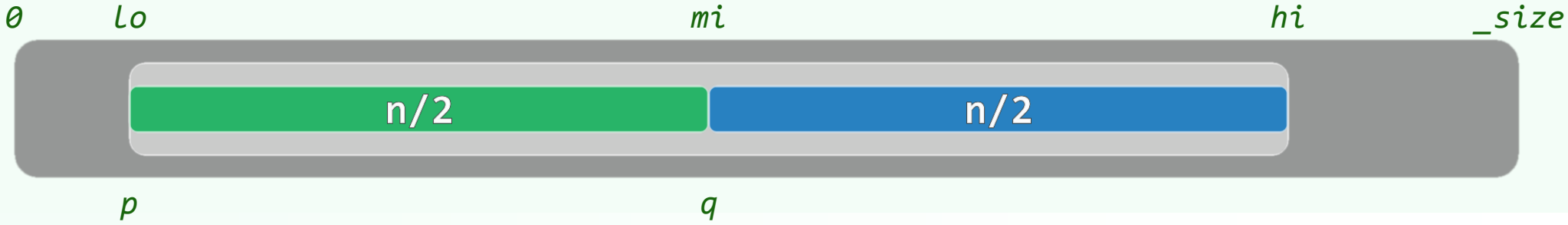
邓俊辉

deng@tsinghua.edu.cn

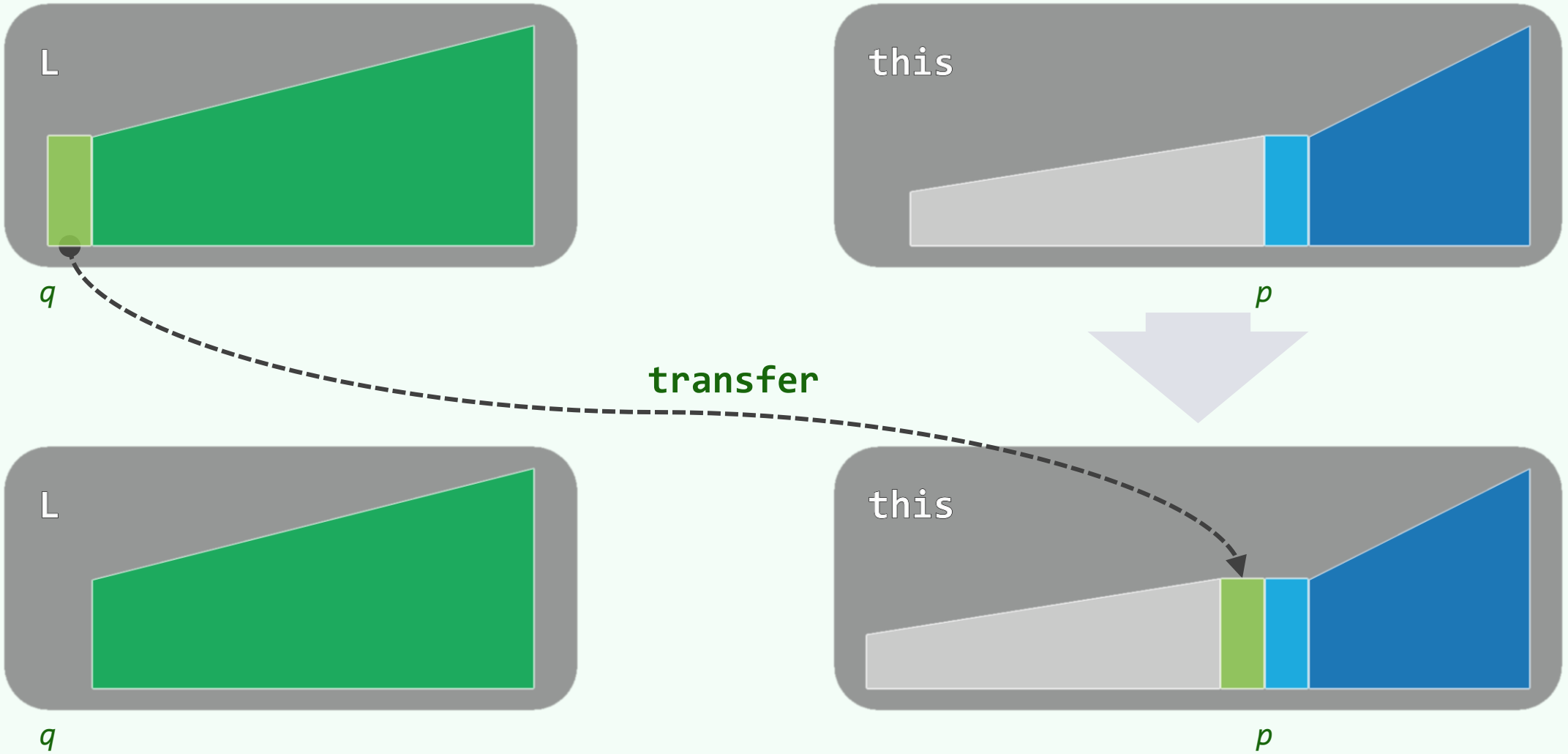
曰两美其必合兮，孰信修而慕之？思九州岛之博大兮，岂惟是其有女？

主算法

```
❖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：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> //自p起的n个元素，与L中自q起的m个元素归并（归并排序时L==this）
Posi(T) List<T>::merge( Posi(T) p, int n, List<T> & L, Posi(T) q, int m ) {
   Posi(T) pp = p->pred; //归并之后p可能不再指向首节点，故需先记忆，以便在返回前更新
   while ( ( 0 < m ) && ( q != p ) ) //q尚未出界（或在mergeSort()中，p尚未出界）之前
      if ( ( 0 < n ) && ( p->data <= q->data ) ) //若p尚未出界且v(p) <= v(q)，则
         { if ( q == (p = p->succ) ) break; n--; } //p直接后移，即完成归入
      else //否则，将q转移至p之前，以完成归入
         { insertB( p, L.remove( ( q = q->succ )->pred ) ); m--; }
   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
```