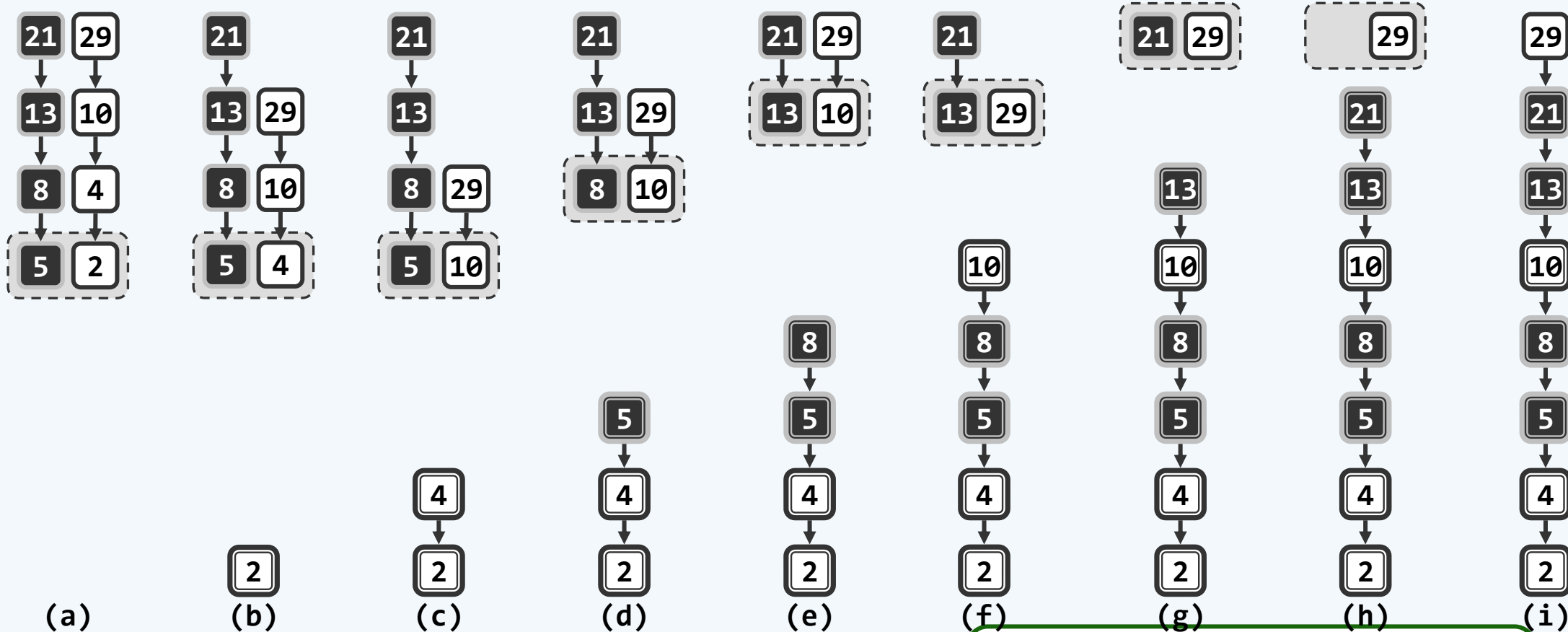θ2-F2

2.向量

归并排序

二路归并

邓俊辉

天下大势，分久必合，合久必分

deng@tsinghua.edu.cn

二路归并
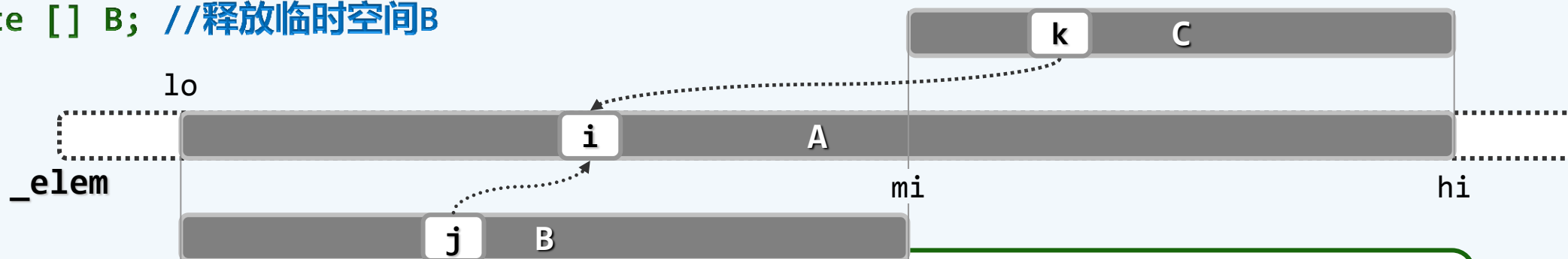
❖ 2-way merge：两个有序序列，合并为一个有序序列：

S[lo, hi) = S[lo, mi) + S[mi, hi)
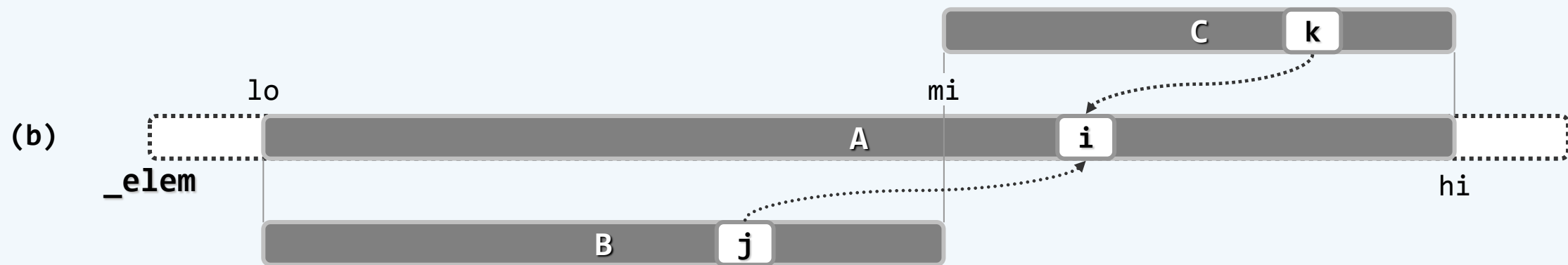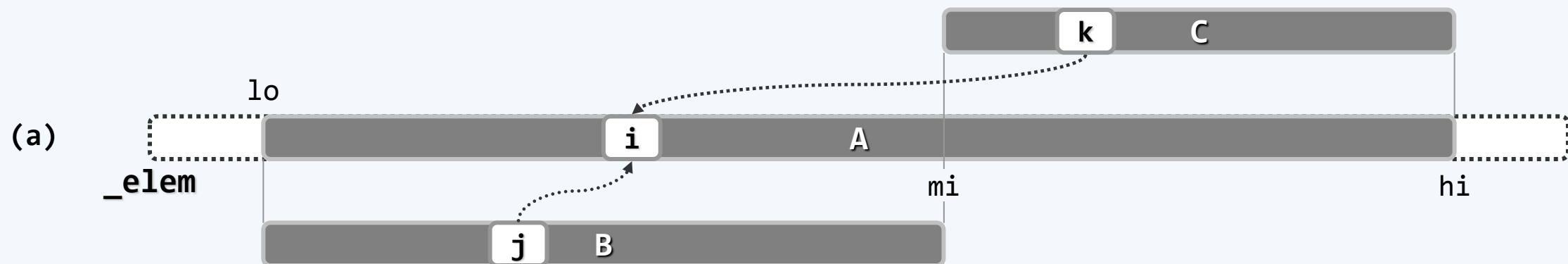
```
template <typename T> void Vector<T>::merge( Rank lo, Rank mi, Rank hi ) {

    T* A = _elem + lo; int lb = mi - lo; T* B = new T[lb]; //A[0, hi - lo) = _elem[lo, hi)

    for ( Rank i = 0; i < lb; B[i] = A[i++] ); //复制前子向量B[0, lb) = _elem[lo, mi)

    int lc = hi - mi; T* C = _elem + mi; //后子向量C[0, lc) = _elem[mi, hi)

    for ( Rank i = 0, j = 0, k = 0; j < lb || k < lc; ) { //B[j]和C[k]中小者 转至A的末尾

        if ( j < lb && ( lc <= k || B[j] <= C[k] ) ) A[i++] = B[j++]; //C[k]已无或不小

        if ( k < lc && ( lb <= j || C[k] < B[j] ) ) A[i++] = C[k++]; //B[j]已无或更大

    } //该循环实现紧凑；但就效率而言，不如拆分处理

    delete [] B; //释放临时空间B

}
```

正确性

(a)

```
                                         k     C

      lo
                     i              A
_elem                                    mi            hi
             j   B
```

(b)

```
                                    C     k

                       mi
      lo
                        A        i
_elem                                          hi
          B      j
```

正确性

(c)

(d)

精简实现

❖   `for ( Rank i = 0, j = 0, k = 0; (j < lb) || (k < lc); ) {`

     `if ( k < lc && ( lb <= j || C[k] <  B[j] ) ) A[i++] = C[k++];`

      `if ( j < lb && ( lc <= k || B[j] <= C[k] ) ) A[i++] = B[j++];`

  `} //交换循环体内两句的次序，删除冗余逻辑`