

向量

归并排序：二路归并

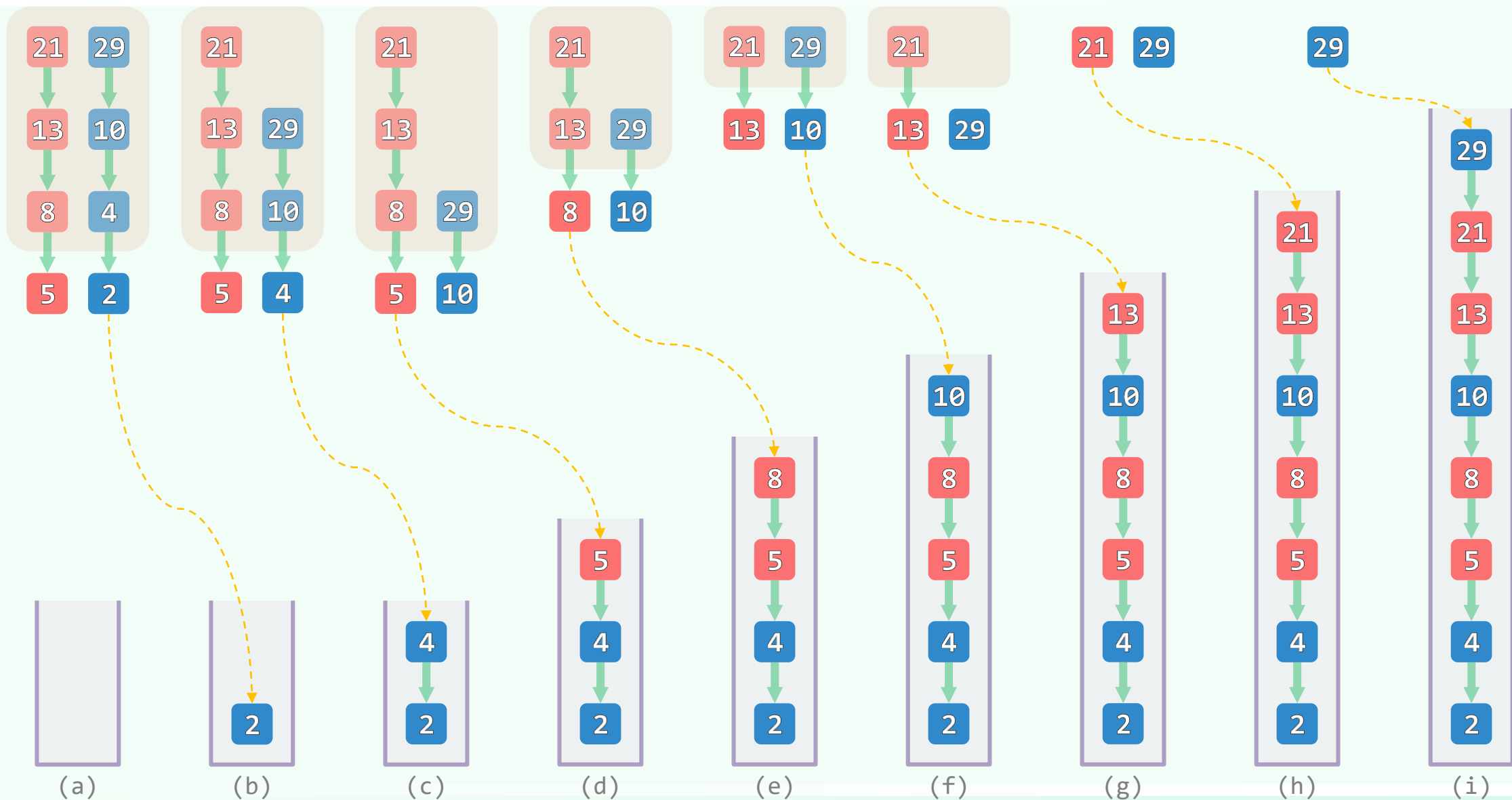
Θ_2-F_2

天下大势，分久必合，合久必分

邓俊辉

deng@tsinghua.edu.cn

2-way merge: 有序序列合二为一, 保持有序: $S[lo, hi) = S[lo, mi) + S[mi, hi)$



实现 (1/2) : 预备

```
template <typename T> //[lo, mi)和[mi, hi)各自有序
```

```
void Vector<T>::merge( Rank lo, Rank mi, Rank hi ) { //lo < mi < hi
```

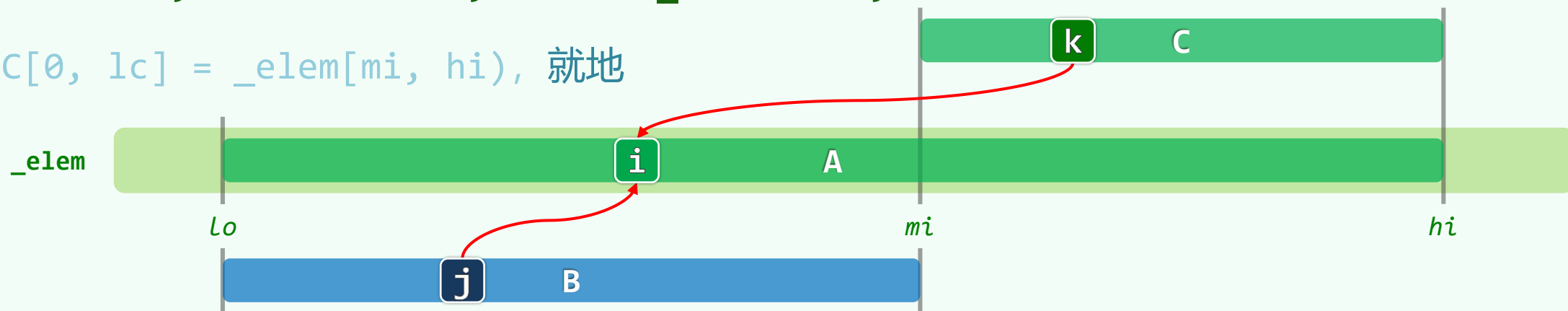
```
    Rank i = 0; T* A = _elem + lo; //A = _elem[lo, hi), 就地
```

```
    Rank j = 0, lb = mi - lo; T* B = new T[lb]; //B[0, lb) <-- _elem[lo, mi)
```

```
    for ( Rank i = 0; i < lb; i++ ) B[i] = A[i]; //复制自A的前缀
```

```
    Rank k = 0, lc = hi - mi; T* C = _elem + mi;
```

```
    //C[0, lc] = _elem[mi, hi), 就地
```



实现 (2/2) : 归并

```
while ( ( j < lb ) && ( k < lc ) ) //反复地比较B、C的首元素
```

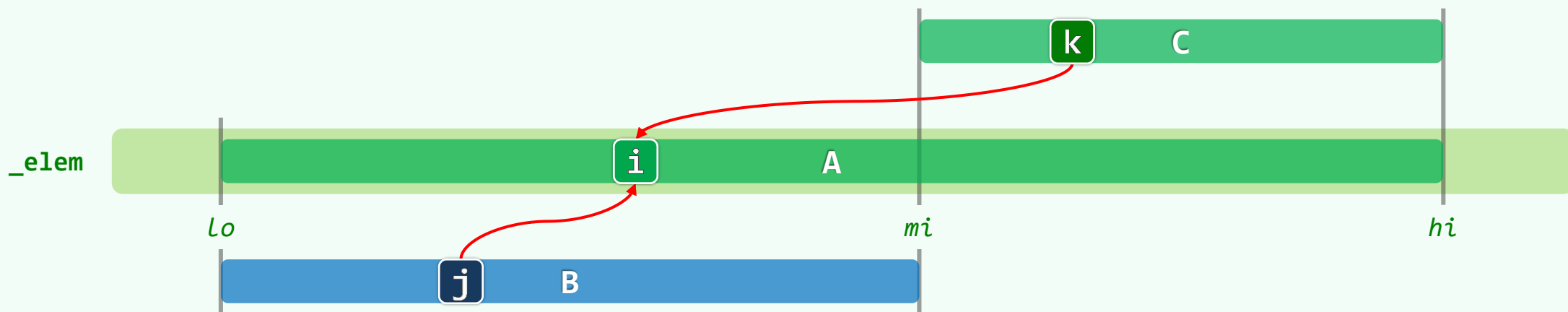
```
    A[i++] = ( B[j] <= C[k] ) ? B[j++] : C[k++]; //小者优先归入A中
```

```
while ( j < lb ) //若C先耗尽, 则
```

```
    A[i++] = B[j++]; //将B残余的后缀归入A中——若B先耗尽呢?
```

```
delete [] B; //new和delete非常耗时, 如何减少?
```

```
}
```



正确性 & 效率

