向量

可扩充向量：算法

02-B1

一个人办一县事，要有一省的眼光；
办一省事，要有一国之眼光；
办一国事，要有世界的眼光。

其实“我”不需扩大，宇宙只是一个“我”，只有在
我们精神往下陷落时，宇宙与我才分开

邓俊辉
deng@tsinghua.edu.cn

静态空间管理

❖ 开辟内部数组_elem[]并使用一段地址连续的物理空间

_capacity：总容量

_size：当前的实际规模n

❖ 若采用静态空间管理策略，容量_capacity固定，则有明显的不足...

❖ - 上溢/overflow：_elem[]不足以存放所有元素，尽管此时系统往往仍有足够的空间

- 下溢/underflow：_elem[]中的元素寥寥无几

- 装填因子/load factor：λ = _size/_capacity << 50%

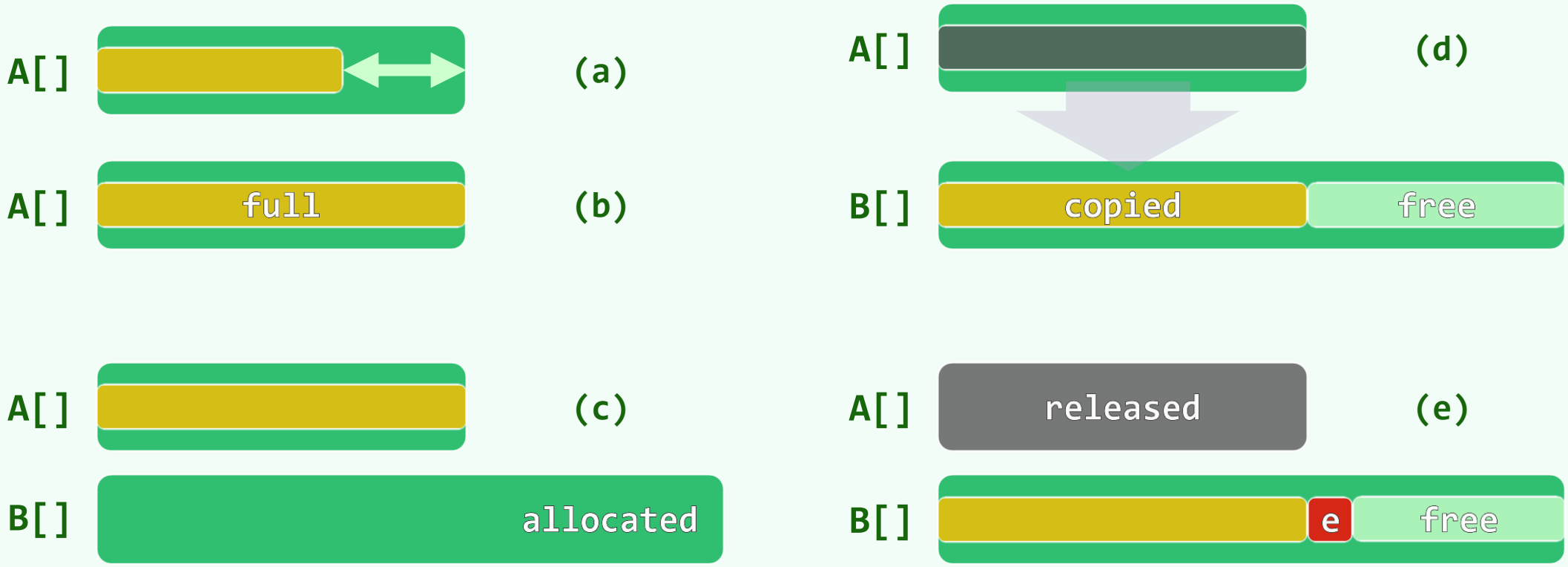
❖ 更糟糕的是，一般的应用环境中难以准确预测空间的需求量

❖ 可否使得向量可随实际需求动态调整容量，并同时保证高效率？

动态空间管理

❖ 蝉的策略：身体经过一段时间的生长，会蜕去原先的外壳，代之以更大的新外壳

❖ 向量：在即将上溢时，适当扩大内部数组的容量



扩容算法

```
template <typename T> void Vector<T>::expand() { //向量空间不足时扩容
    if ( _size < _capacity ) return; //尚未满员时，不必扩容
    _capacity = max( _capacity, DEFAULT_CAPACITY ); //不低于最小容量
    T* oldElem = _elem; _elem = new T[ _capacity <<= 1 ];//容量加倍
    for ( int i = 0; i < _size; i++ ) //复制原向量内容
       _elem[i] = oldElem[i]; //T为基本类型，或已重载赋值操作符'='
    delete [] oldElem; //释放原空间
} //得益于向量的封装，尽管扩容之后数据区的物理地址有所改变，却不致出现野指针
```

- 为何必须采用容量加倍策略呢？其它策略是否可行？