

向量

位图：快速初始化

02-G3

邓俊辉

deng@tsinghua.edu.cn

我从那无比圣洁的河水那里
走了回来，仿佛再生了一般
正如新的树用新的枝叶更新
一身洁净，准备就绪，就飞往星辰

兵甲之符，右在君，左在杜。凡兴土披甲，用兵五十人以上，必会君符，乃敢行之

$$O(n) \sim O(1)$$

❖ Bitmap的构造函数中，通过 `memset(M,0,N)` 统一清零

这一步只需 $O(1)$ 时间？不，实际上仍等效于诸位清零， $O(N) = O(n)$ ！

❖ 尽管这并不会影响上例的渐近复杂度，但并非所有问题都是如此

❖ 有时，对于大规模的散列表（第09章），初始化的效率直接影响到**实际性能**

例如：第13章中bc[]表的构造算法，需要 $O(|\Sigma| + m) = O(s + m)$ 时间

若能省去bc[]表各项的初始化，则可严格地保证是 $O(m)$

❖ 有时，甚至会影响到算法的**整体**渐近复杂度

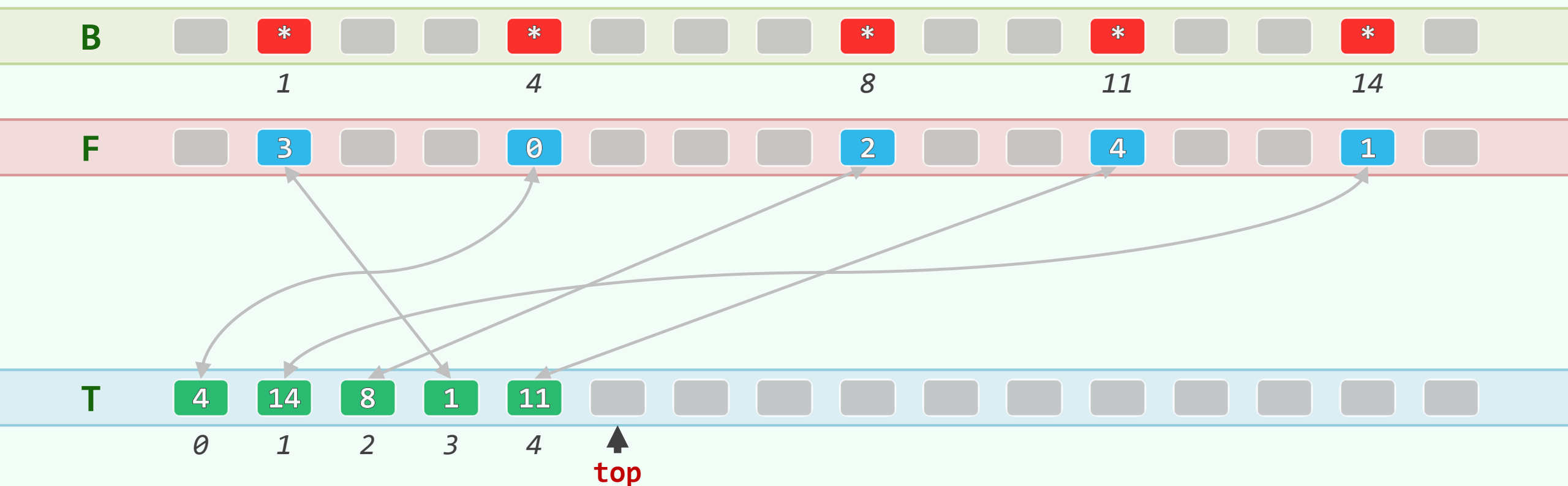
例如，为从 $n = 10^8$ 个 32 位整数中找出**相等者**，可仿造**剔除算法**... //但这里无需回收

因此，若能省去Bitmap的初始化，则只需 $O(n)$ 时间

J. Hopcroft, 1974: **B[]**拆分成一对等长向量: Rank **F[m]**, **T[m]**, **top** = 0;

❖ 构成校验环: $T[F[k]] == k$ & $F[T[i]] == i$

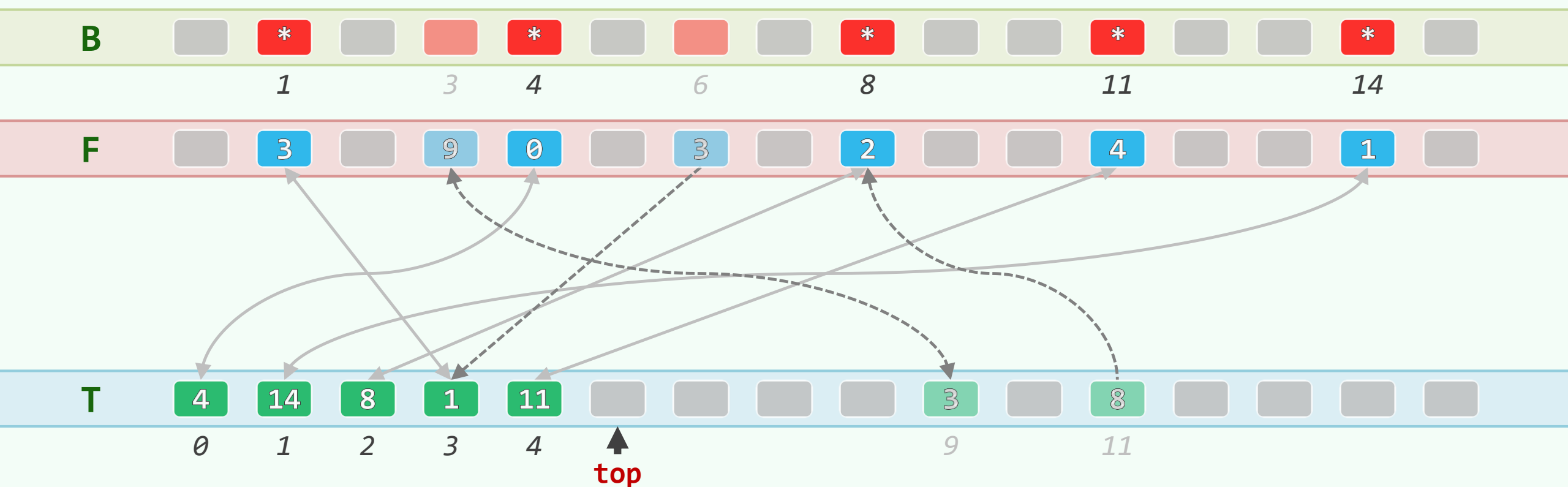
//君右杜左, 虎符会合



测试: $\text{test}(1, 4, 8, 11, 14) = \text{true}$ & $\text{test}(3, 6) = \text{false}$

```
bool Bitmap::test( Rank k )
```

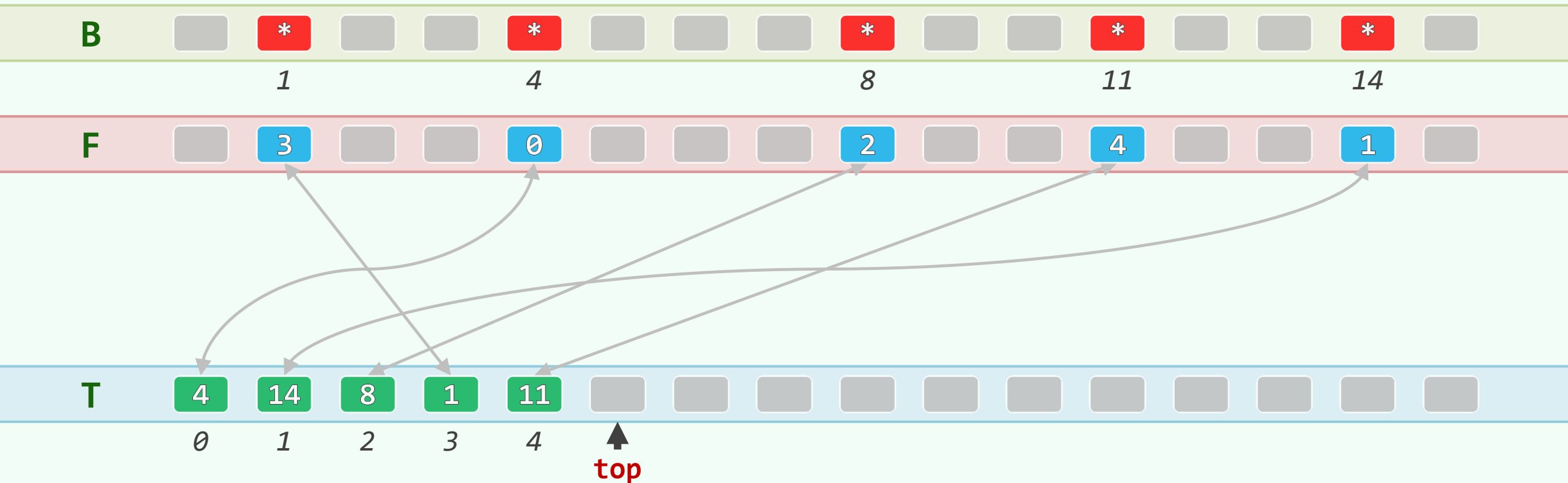
```
{ return (-1 != F[k]) && (F[k] < top) && (k == T[F[k]]); }
```



$O(1)$ 复位

```
void Bitmap::reset()
```

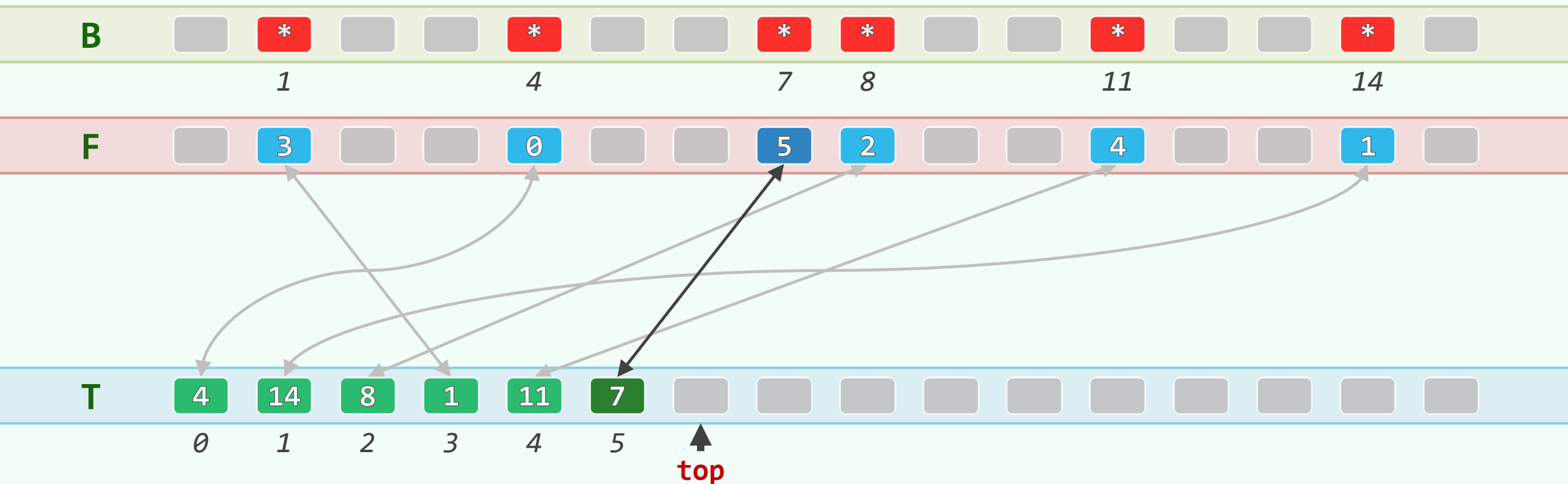
```
{ top = 0; }
```



$O(1)$ 插入: set(7)

```
void Bitmap::set( Rank k )
```

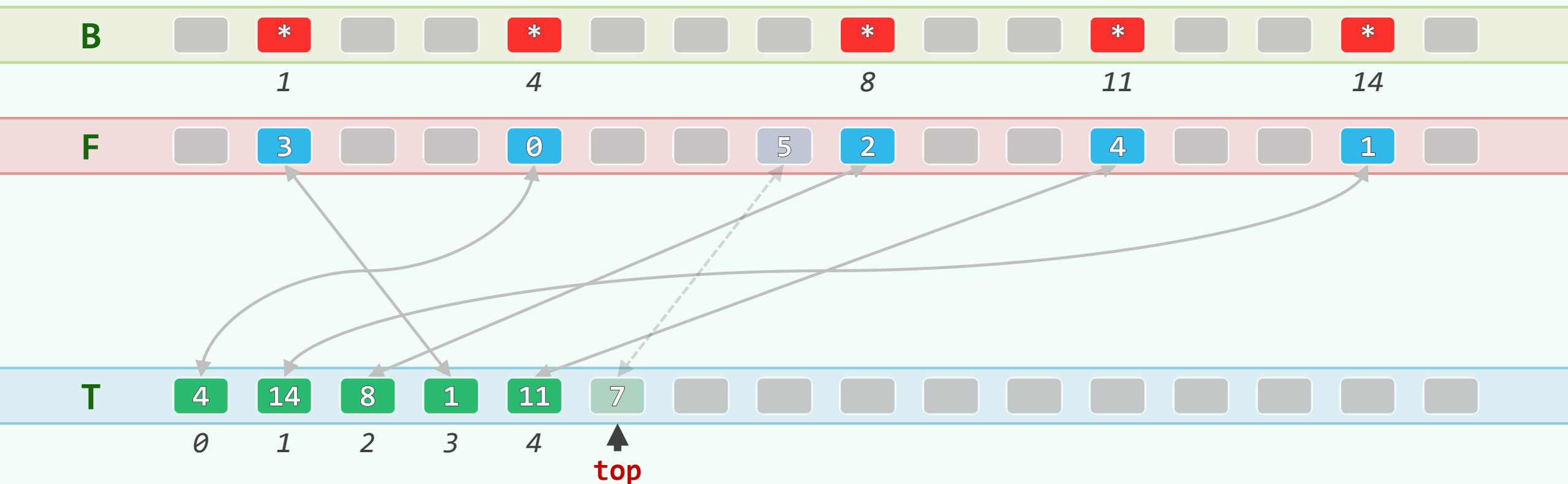
```
{ if ( !test( k ) ) { T[top] = k; F[k] = top++; } }
```



$O(1)$ 删除: remove(7)

```
void Bitmap::clear( Rank k )
```

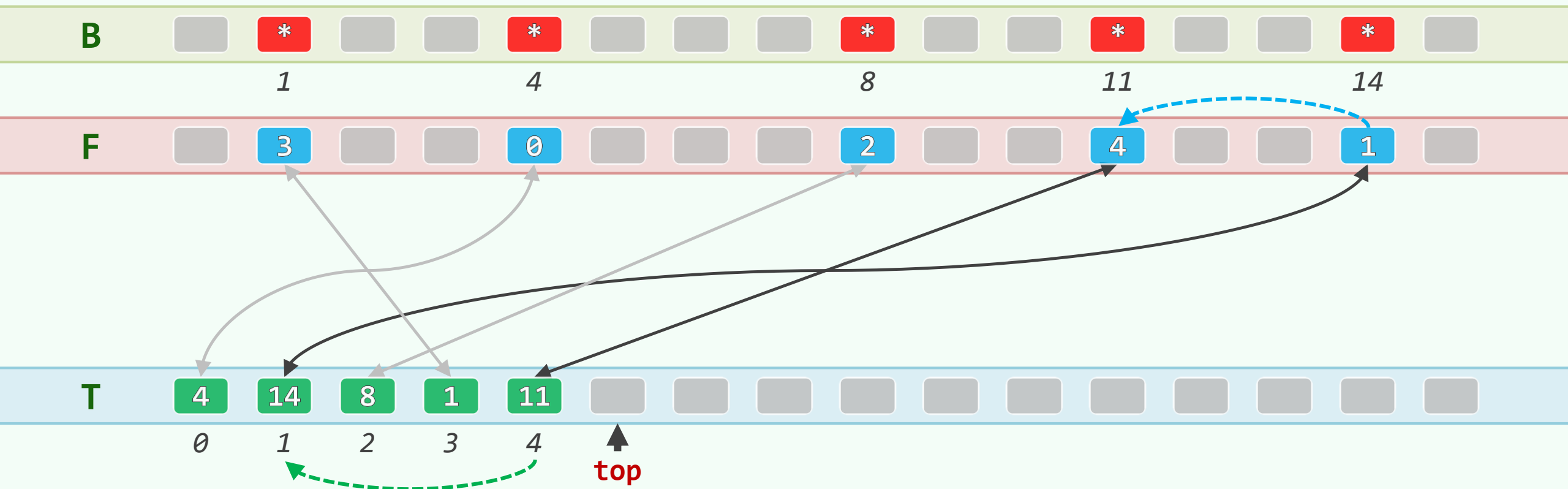
```
{ if ( test( k ) && ( --top ) ) { F[T[top]] = F[k]; T[F[k]] = T[top]; } }
```



$O(1)$ 删除: remove(14) ...

```
void Bitmap::clear( Rank k )
```

```
{ if ( test( k ) && ( --top ) ) { F[T[top]] = F[k]; T[F[k]] = T[top]; } }
```



$O(1)$ 删除: ... remove(14)

```
void Bitmap::clear( Rank k )
```

```
{ if ( test( k ) && ( --top ) ) { F[T[top]] = F[k]; T[F[k]] = T[top]; } }
```

