**一. 概述**

**都是针对已序区间执行的算法。**

|  |  |
| --- | --- |
| binary\_search() | 判断某区间内是否包含某个元素 |
| includes() | 判断某区间内的每一个元素是否都涵盖于另一区间中 |
| lower\_bound() | 搜索第一个"大于等于给定值"的元素 |
| upper \_bound() | 搜索第一个"大于给定值"的元素 |
| equal\_range() | 返回"等于给定值"的所有元素构成的区间 |
| merge() | 将两个区间合并 |
| set\_union() | 求两个区间的并集 |
| set\_intersection() | 求两个区间的交集 |
| set\_difference() | 求位于第一区间但不位于第二区间的所有元素，形成一个已序区间 |
| set\_symmetric\_difference() | 找出只出现于两区间之一的所有元素，形成一个已序区间 |
| inplace\_merge() | 将两个连续的已序区间合并 |

**二. 搜索元素**

**1. 检查某个元素是否存在**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331866)

1. //判断已序区间[beg, end)是否包含和value等值的元素
2. **bool**
3. binary\_search (ForwardIterator beg, ForwardIterator end,
4. const T& value)
5. **bool**
6. binary\_search (ForwardIterator beg, ForwardIterator end,
7. const T& value,
8. BinaryPredicate op)

//判断已序区间[beg, end)是否包含和value等值的元素

bool

binary\_search (ForwardIterator beg, ForwardIterator end,

const T& value)

bool

binary\_search (ForwardIterator beg, ForwardIterator end,

const T& value,

BinaryPredicate op)

**2. 检查若干个值是否存在**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331866)

1. //判断已序区间[beg, end)是否包含另一个已序区间[searchBeg, searchEnd)
2. **bool**
3. includes (InputIterator1 beg, InputIterator1 end,
4. InputIterator2 searchBeg, InputIterator2 searchEnd)
5. **bool**
6. includes (InputIterator1 beg, InputIterator1 end,
7. InputIterator2 searchBeg, InputIterator2 searchEnd,
8. BinaryPredicate op)

//判断已序区间[beg, end)是否包含另一个已序区间[searchBeg, searchEnd)

bool

includes (InputIterator1 beg, InputIterator1 end,

InputIterator2 searchBeg, InputIterator2 searchEnd)

bool

includes (InputIterator1 beg, InputIterator1 end,

InputIterator2 searchBeg, InputIterator2 searchEnd,

BinaryPredicate op)

**3. 搜索第一个或最后一个可能位置**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331866)

1. //返回第一个"大于等于value"的元素位置，如果不存在返回end
2. ForwardIterator
3. lower\_bound (ForwardIterator beg, ForwardIterator end,
4. const T& value)
5. ForwardIterator
6. lower\_bound (ForwardIterator beg, ForwardIterator end,
7. const T& value,
8. BinaryPredicate op)
9. //返回第一个"大于value"的元素位置
10. ForwardIterator
11. upper\_bound (ForwardIterator beg, ForwardIterator end,
12. const T& value)
13. ForwardIterator
14. upper\_bound (ForwardIterator beg, ForwardIterator end,
15. const T& value,
16. BinaryPredicate op)

//返回第一个"大于等于value"的元素位置，如果不存在返回end

ForwardIterator

lower\_bound (ForwardIterator beg, ForwardIterator end,

const T& value)

ForwardIterator

lower\_bound (ForwardIterator beg, ForwardIterator end,

const T& value,

BinaryPredicate op)

//返回第一个"大于value"的元素位置

ForwardIterator

upper\_bound (ForwardIterator beg, ForwardIterator end,

const T& value)

ForwardIterator

upper\_bound (ForwardIterator beg, ForwardIterator end,

const T& value,

BinaryPredicate op)

**4. 搜索第一个和最后一个可能位置**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331866)

1. //返回"与value相等"的元素所形成的区间
2. pair<ForwardIterator,ForwardIterator>
3. equal\_range (ForwardIterator beg, ForwardIterator end,
4. const T& value)
5. pair<ForwardIterator,ForwardIterator>
6. equal\_range (ForwardIterator beg, ForwardIterator end,
7. const T& value,
8. BinaryPredicate op)

//返回"与value相等"的元素所形成的区间

pair<ForwardIterator,ForwardIterator>

equal\_range (ForwardIterator beg, ForwardIterator end,

const T& value)

pair<ForwardIterator,ForwardIterator>

equal\_range (ForwardIterator beg, ForwardIterator end,

const T& value,

BinaryPredicate op)

**三. 合并元素**

**1. 两个已序集合的总合(Sum)**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331866)

1. //source1: 1 2 2 4 6 7 7 9
2. //source2: 2 2 2 3 6 6 8 9
3. //Sum: 1 2 2 2 2 2 3 4 6 6 6 7 7 8 9 9
4. //将源区间[source1Beg, source1End)和[source2Beg, source2End)内的元素合并
5. //以destBeg为起点输出
6. OutputIterator
7. merge (InputIterator source1Beg, InputIterator source1End,
8. InputIterator source2Beg, InputIterator source2End,
9. Output Iterator destBeg)
10. OutputIterator
11. merge (InputIterator source1Beg, InputIterator source1End,
12. InputIterator source2Beg, InputIterator source2End,
13. OutputIterator destBeg,
14. BinaryPredicate op)

//source1: 1 2 2 4 6 7 7 9

//source2: 2 2 2 3 6 6 8 9

//Sum: 1 2 2 2 2 2 3 4 6 6 6 7 7 8 9 9

//将源区间[source1Beg, source1End)和[source2Beg, source2End)内的元素合并

//以destBeg为起点输出

OutputIterator

merge (InputIterator source1Beg, InputIterator source1End,

InputIterator source2Beg, InputIterator source2End,

Output Iterator destBeg)

OutputIterator

merge (InputIterator source1Beg, InputIterator source1End,

InputIterator source2Beg, InputIterator source2End,

OutputIterator destBeg,

BinaryPredicate op)

**2.. 两个已序集合的并集(Union)**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331866)

1. //source1: 1 2 2 4 6 7 7 9
2. //source2: 2 2 2 3 6 6 8 9
3. //Union: 1 2 2 2 3 4 6 6 7 7 8 9
4. //这个区间内含的元素要不来自第一源区间，要不来自第二源区间
5. OutputIterator
6. set\_union (InputIterator source1Beg, InputIterator source1End,
7. InputIterator source2Beg, InputIterator source2End,
8. OutputIterator destBeg)
9. OutputIterator
10. set\_union (InputIterator source1Beg, InputIterator source1End,
11. InputIterator source2Beg, InputIterator source2End,
12. OutputIterator destBeg,
13. BinaryPredicate op)

//source1: 1 2 2 4 6 7 7 9

//source2: 2 2 2 3 6 6 8 9

//Union: 1 2 2 2 3 4 6 6 7 7 8 9

//这个区间内含的元素要不来自第一源区间，要不来自第二源区间

OutputIterator

set\_union (InputIterator source1Beg, InputIterator source1End,

InputIterator source2Beg, InputIterator source2End,

OutputIterator destBeg)

OutputIterator

set\_union (InputIterator source1Beg, InputIterator source1End,

InputIterator source2Beg, InputIterator source2End,

OutputIterator destBeg,

BinaryPredicate op)

**3. 两个已序集合的交集(Intersection)**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331866)

1. //source1: 1 2 2 4 6 7 7 9
2. //source2: 2 2 2 3 6 6 8 9
3. //Inter: 2 2 6 9
4. //元素必须同时在两个区间
5. OutputIterator
6. set\_intersection (InputIterator source1Beg, InputIterator source1End,
7. InputIterator source2Beg, InputIterator source2End,
8. OutputIterator destBeg)
9. OutputIterator
10. set\_intersection (InputIterator source1Beg, InputIterator source1End,
11. InputIterator source2Beg, InputIterator sotirce2End,
12. OutputIterator destBeg,
13. BinaryPredicate op)

//source1: 1 2 2 4 6 7 7 9

//source2: 2 2 2 3 6 6 8 9

//Inter: 2 2 6 9

//元素必须同时在两个区间

OutputIterator

set\_intersection (InputIterator source1Beg, InputIterator source1End,

InputIterator source2Beg, InputIterator source2End,

OutputIterator destBeg)

OutputIterator

set\_intersection (InputIterator source1Beg, InputIterator source1End,

InputIterator source2Beg, InputIterator sotirce2End,

OutputIterator destBeg,

BinaryPredicate op)

**4. 两个已序集合的差集(Difference)**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331866)

1. //source1: 1 2 2 4 6 7 7 9
2. //source2: 2 2 2 3 6 6 8 9
3. //differ: 1 4 7 7
4. //只在第一个区间，不在第二个区间
5. OutputIterator
6. set\_difference (InputIterator source1Beg, InputIterator source1End,
7. InputIterator source2Beg, InputIterator source2End,
8. OutputIterator destBeg)
9. OutputIterator
10. set\_difference (InputIterator source1Beg, InputIterator source1End,
11. InputIterator source2Beg, InputIterator source2End,
12. OutputIterator destBeg,
13. BinaryPredicate op)

//source1: 1 2 2 4 6 7 7 9

//source2: 2 2 2 3 6 6 8 9

//differ: 1 4 7 7

//只在第一个区间，不在第二个区间

OutputIterator

set\_difference (InputIterator source1Beg, InputIterator source1End,

InputIterator source2Beg, InputIterator source2End,

OutputIterator destBeg)

OutputIterator

set\_difference (InputIterator source1Beg, InputIterator source1End,

InputIterator source2Beg, InputIterator source2End,

OutputIterator destBeg,

BinaryPredicate op)

**5. 将连贯的已序区间合并**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331866)

1. //将已序[beg1, end1beg2)和[end1beg2, end2)的元素合并，
2. //使区间[beg1, end2)成为两者之和
3. void
4. inplace\_merge (BidirectionalIterator beg1,
5. BidirectionalIterator end1beg2,
6. BidirectionalIterator end2)
7. void
8. inplace\_merge (BidirectionalIterator beg1,
9. BidirectionalIterator end1beg2,
10. BidirectionalIterator end2,
11. BinaryPredicate op)