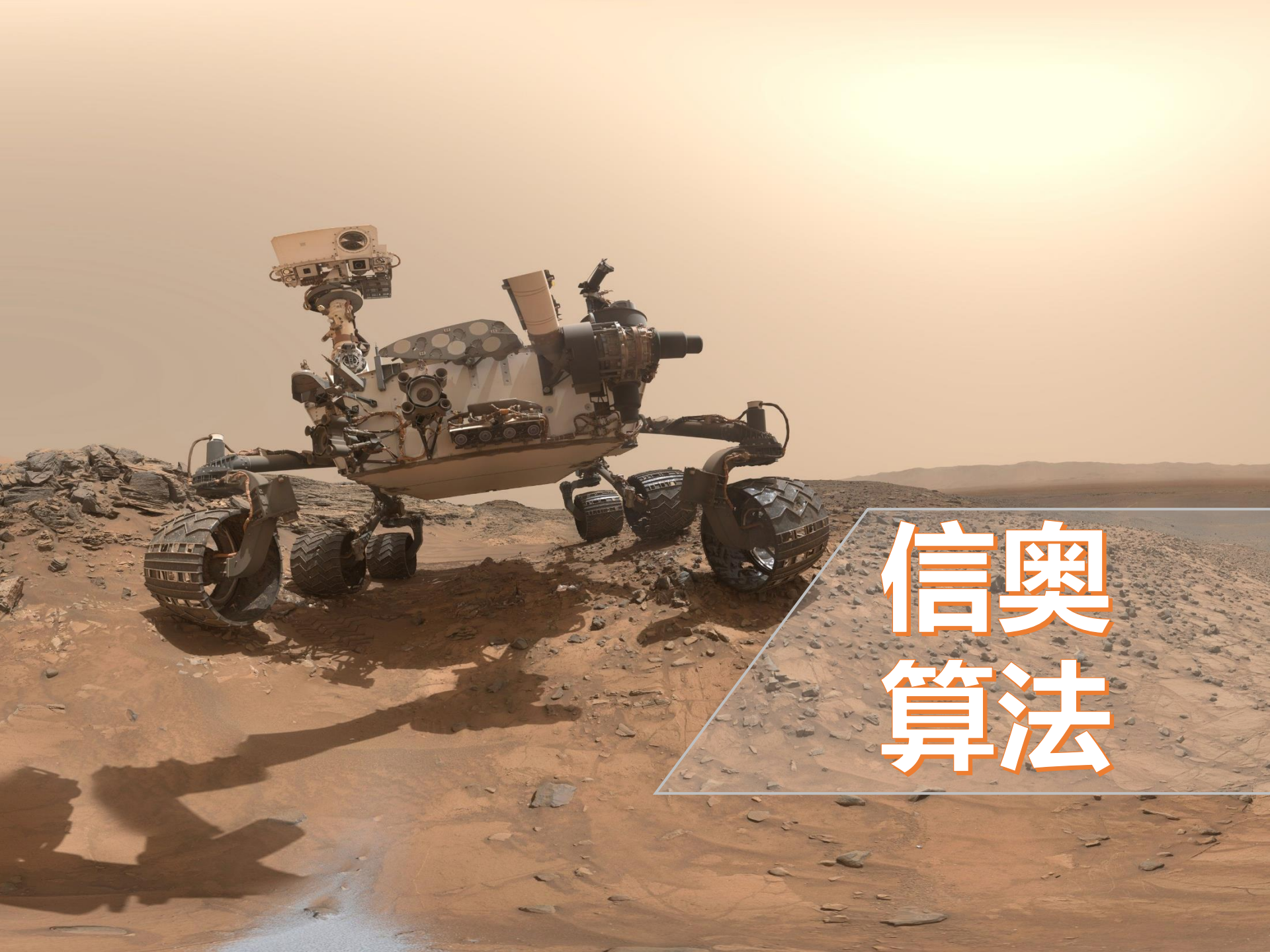
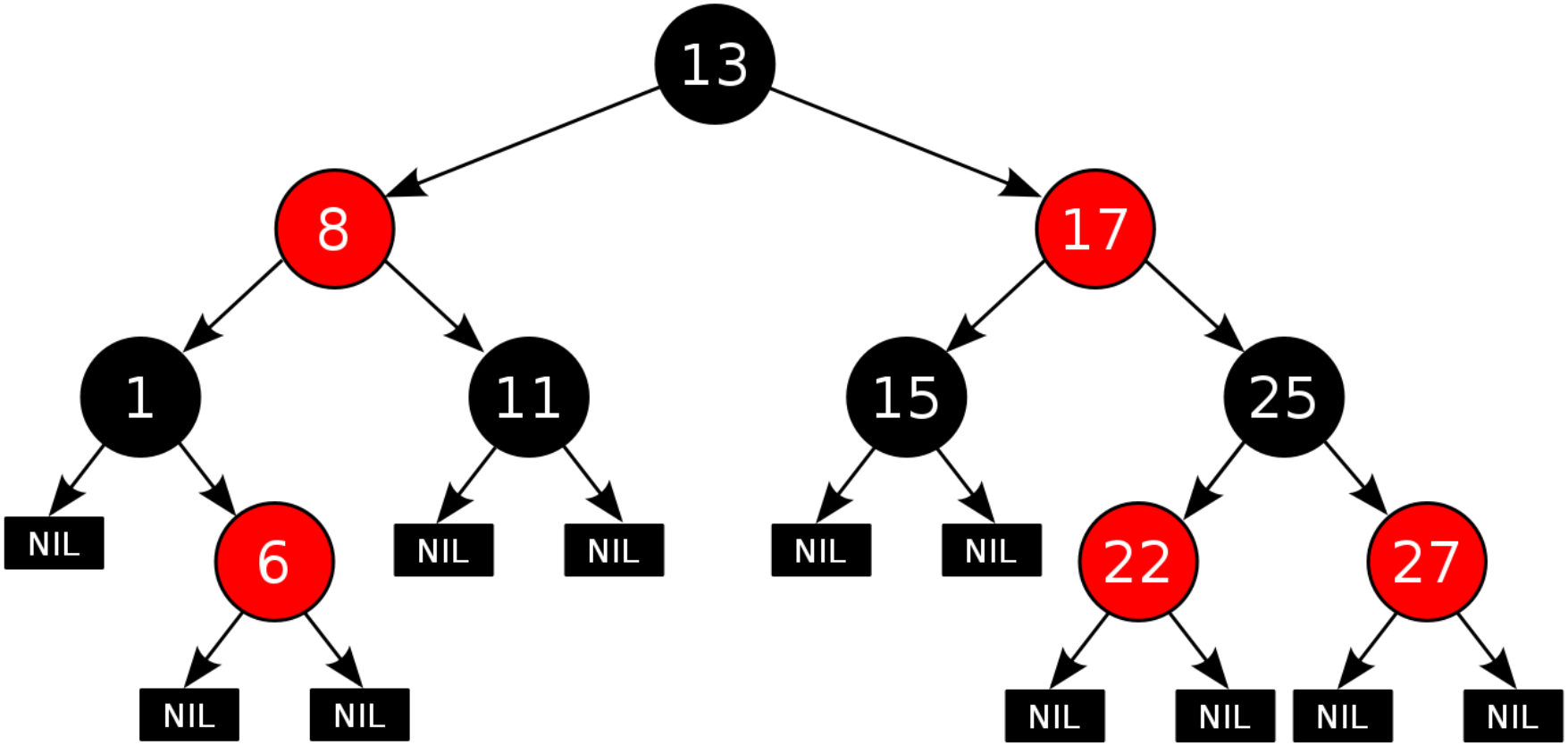

信奥
算法

数据**容器**

multiset

set

红黑树：一种平衡二叉查找树



set和multiset的底层实现都是红黑树

易错点

元素应该**重复**出现时，
不能使用set，
应该使用multiset

大部分情况，
建议使用multiset

数据容器：multiset和set

multiset 和 set 可较快完成对一组数据的常规操作，包括：

插入　删除

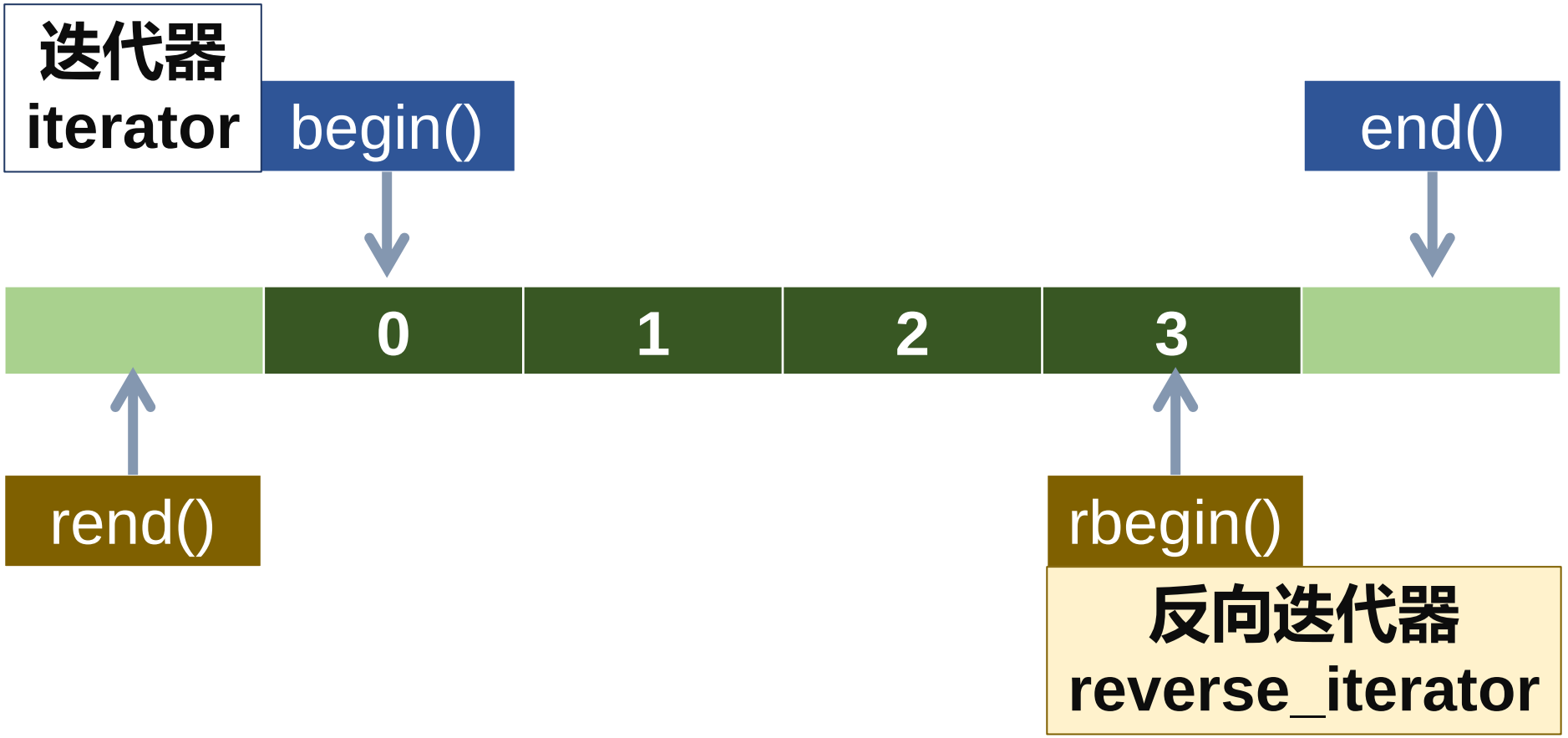
查找　计数　去重

排序　找最小/最大

multiset 允许元素重复出现

set 保证元素唯一性，不允许元素重复

begin(),end(),rbegin(),rend()



迭代器 和 反向迭代器

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(8); s.insert(7);
7      s.insert(1); s.insert(2); s.insert(3);
8      multiset<int>::iterator it;
9      for(it=s.begin();it!=s.end();it++)
10         cout<<*it;
11     cout<<endl<<"------"<<endl;
12     multiset<int>::reverse_iterator rit;
13     for(rit=s.rbegin();rit!=s.rend();rit++)
14         cout<<*rit;
15     return 0;
16 }
```

找最小值

begin()

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(8); s.insert(8);
7      s.insert(6); s.insert(6);
8      int smallest=*(s.begin());
9      cout<<smallest<<endl;
10     return 0;
11 }
```

*(s.begin())

找第二小元素: 方法1

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(8);
7      s.insert(7); s.insert(6);
8      s.erase(s.begin());
9      int ans=*(s.begin());
10     cout<<ans<<endl;
11     return 0;
12 }
```

删除最小元素

s.erase(s.begin())

*(s.begin())

找第二小元素: 方法2

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(8);
7      s.insert(7); s.insert(6);
8      multiset<int>::iterator it;
9      it=s.begin();  it++;
10     cout<<*it<<endl;
11     return 0;
12 }
```

迭代器往后移

begin()

it++

*it

找第二小元素: 易错点

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(8);
7      s.insert(6); s.insert(6);
8      int x=*(s.begin());
9      s.erase(x);
10     int ans=*(s.begin());
11     cout<<ans<<endl;
12     return 0;
13 }
```

会删除所有等于最小值元素

找最大值:方法1 rbegin()

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(8); s.insert(8);
7      s.insert(6); s.insert(6);
8      int biggest=*(s.rbegin());
9      cout<<biggest<<endl;
10     return 0;
11 }
```

反向迭代器

*(s.rbegin())

找最大值:方法2

end()

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(8); s.insert(8);
7      s.insert(6); s.insert(6);
8      multiset<int>::iterator it;
9      it=s.end(); it--;
10     cout<<*it<<endl;
11     return 0;
12 }
```

迭代器前移

end()

it--

找第二大元素: 方法1

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(8);
7      s.insert(7); s.insert(6);
8      multiset<int>::iterator it;
9      it=s.end();  it--; it--;
10     cout<<*it<<endl;
11     return 0;
12 }
```

迭代器前移

end()

it--

*it

找第二大元素: 方法2

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(8);
7      s.insert(7); s.insert(6);
8      multiset<int>::iterator it;
9      it=s.end();  it--; //指向目前最大元素
10     s.erase(it);
11     it=s.end();  it--; //指向目前最大元素
12     cout<<*it<<endl;
13     return 0;
14 }
```

迭代器前移

只删除一个最大元素

找第二大元素: 方法3

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(8);
7      s.insert(7); s.insert(6);
8      multiset<int>::reverse_iterator it;
9      it=s.rbegin();  it++;
10     cout<<*it<<endl;
11     return 0;
12 }
```

反向迭代器移动

rbegin()

it++

*it

找第二大元素: 易错点1

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(9);
7      s.insert(7); s.insert(6);
8      int x=*(s.rbegin());
9      s.erase(x);
10     int ans=*(s.rbegin());
11     cout<<ans<<endl;
12     return 0;
13 }
```

会删除所有等于最大值元素

找第二大元素: 易错点2

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(9);
7      s.insert(7); s.insert(6);
8      s.erase(s.rbegin());//报错
9      int ans=*(s.rbegin());
10     cout<<ans<<endl;
11     return 0;
12 }
```

erase()参数不可用反向迭代器

s.erase(s.rbegin())会报错

找第二大元素: 易错点3

```
1  #include<iostream>
2  #include<set>
3  using namespace std;
4  int main() {
5      multiset<int> s;
6      s.insert(9); s.insert(8);
7      s.insert(7); s.insert(6);
8      multiset<int>::iterator it;
9      it=s.end(); it--;
10     s.erase(it); it--;
11     cout<<*it<<endl;
12     return 0;
13 }
```

删除元素后迭代器会失效

2637

数字合并

定义一个multiset容器叫s
存放所有目前的数字

注意：
可能有重复

贪心算法选数字，重复执行：

每次找最大的两个数字a和b合并。
也就是从容器s中取出最大两个数a和b，
然后删除最大的两个数，
再插入a*b+1

```
 6      multiset<int> s;
 7      multiset<int>::iterator it;
 8      int n;
 9      cin>>n;
10      for(int i=0;i<n;i++) {
11          int x;
12          cin>>x;
13          s.insert(x);
14      }
```

```
15  while(                    ){
16      it=s.end();
17      it--;
18      int a=*it;
19      
20      it=s.end();
21      it--;
22      int b=*it;
23      s.erase(it);
24      
25  }
26  cout<<                    <<endl;
```

113

数字合并

定义一个multiset容器叫s
存放所有目前的数字

注意：
可能有重复

贪心算法选数字，重复执行：

每次找最小的两个数字a和b合并。
也就是从容器s中取出最小两个数a和b，
然后删除最小的两个数，
再插入a+b
总费用累加a+b

51

接水问题

定义一个multiset容器叫s
存放所有龙头当前的使用者的结束时间

注意：
可能有重复

模拟接水过程

初始化：m个龙头当前使用者结束时间为0

依次处理n个人

每次找最早的结束时间t
从容器s中取出最小数t，
然后删除该最小数，
再插入（t+新接水的人的使用时间x)

```
 6    multiset<int> s;
 7    int n,m;
 8    cin>>n>>m;
 9    for(int i=1;i<=m;i++)
10        s.insert(0);
```

```
11  for(int i=1;i<=n;i++){
12      int x;
13      cin>>x;
14      int t=*s.begin();
15      
16      s.insert(t+x);
17  }
18  cout<<          <<endl;
```

是否可以用
s.erase(t)

参考资料

http://www.cplusplus.com/reference/set/set/

http://www.cplusplus.com/reference/set/multiset/

快快编程作业

	2637
	113
	51
拓展题	304