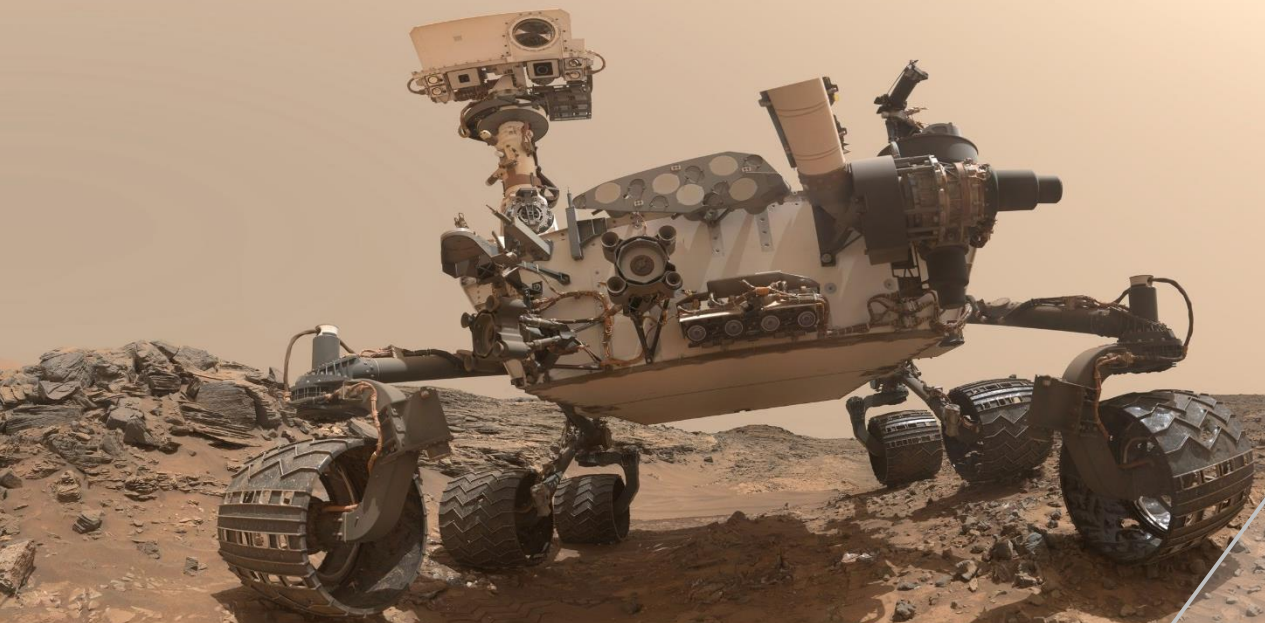
CS101



信奥算法

课件下载地址：
http://pan.baidu.com/s/1nu6kYkL

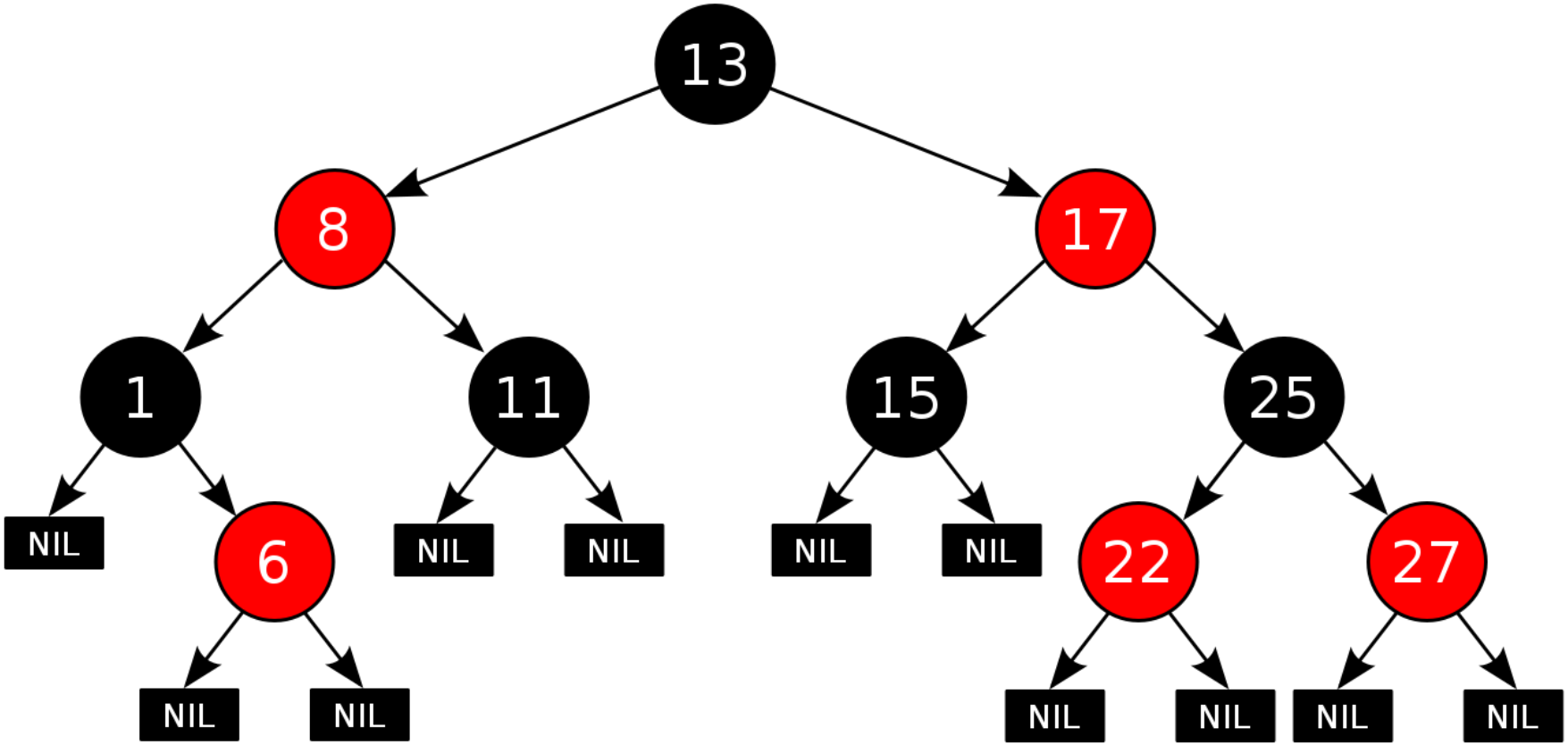
作业网站：
http://120.132.18.213:8080/thrall-web/main#home

数据容器

multiset

set

数据结构：红黑树RB-tree



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

作业1：捡袜子

路边捡垃圾的老头正在收集袜子，每只他捡到的袜子都有一种颜色，他希望将相同颜色的袜子配对成为一双袜子。输入第一行为正整数n表示共有几只袜子，第二行为n个小写的英文单词代表每只袜子的颜色，由空格隔开。输出共有几双袜子配对成功。

输入样例
7
red black gold red red black green

输出样例
2

```cpp
#include<iostream>
#include<string>
#include<set>
using namespace std;
set<string> s;
int main() {
    int ans=0,n,i;
    cin>>n;
    for(i=0;i<n;i++){
        string x;
        cin>>x;
        if(s.count(x)==0) s.insert(x);
        else{
            ans++;
            s.erase(x);
        }
    }
    cout<<ans<<endl;
    return 0;
}
```

作业2：英雄联盟

来自世界各地的英雄人物要组成一个联盟，简称HL。但是HL联盟并不稳定，时常有英雄会因为意见不合离开联盟，也会有新的英雄加入。
输入第一行为正整数m代表有m条关于联盟的信息。以下每一行为加号+或者减号-，以及英雄名字，代表该英雄试图加入或者离开联盟。输出联盟剩余英雄的字典序排列。

输入样例
7
+ironman
+thor
+spiderman
-thor
-thor
-batman
+spiderman

输出样例
ironman
spiderman

```cpp
#include<iostream>
#include<string>
#include<set>
using namespace std;
set<string> s;
set<string>::iterator it;
string x;
int ans,n,i,a,b,c;
int main() {
    cin>>n;
    for(i=0;i<n;i++){
        cin>>x;
        char ch=x[0]; x.erase(0,1);
        if(ch=='+') s.insert(x);
        else s.erase(x);
    }
    for(it=s.begin();it!=s.end();it++)
        cout<<*it<<endl;
    return 0;
}
```

作业3：僵尸传染

僵尸大战爆发了，最近正常人和僵尸人之间发生了很多互相撕咬的事件。一开始只有一头叫做zombie的僵尸人，僵尸病毒传染性太强，如果正常人被僵尸人咬了一口就会变身成僵尸。输入第一行是m代表有m条互咬信息，以下为m行，按照撕咬先后顺序排列，每行有两个人的小写名字，表示他们两人互相撕咬。输出为最终共有多少个僵尸。

输入样例
5
david mike
mike zombie
zhang mike
wang david
shawn david

输出样例
3

```cpp
#include<iostream>
#include<string>
#include<set>
using namespace std;
int main() {
    set<string> s;
    set<string>::iterator it;
    s.insert("zombie");
    int m;
    cin>>m;
    for(int i=0;i<m;i++) {
        string a,b;
        cin>>a>>b;
        if(s.count(a)){
            if(!s.count(b)) s.insert(b);
        }else if(s.count(b)) s.insert(a);
    }
    cout<<s.size()<<endl;
    return 0;
}
```

作业4：电影推荐

电影爱好者佳佳希望根据她喜欢看的电影，找到另外一些电影，也能符合她口味。为此，她需要知道任意两部电影的相似程度。第一步，她把每部电影加上关键词，第二步，比较两部电影的关键词看看有多少是相同的。

输入有两行代表两部电影，每行冒号前是电影名称，冒号后是一个空格，和一些英文单词，代表这部电影的关键词。输出这两部电影的相似程度，用分数形式表示:a/b，其中整数b表示第二部电影有多少个关键词，整数a表示第二部电影有多少个关键词也在第一部电影里面。

输入样例

World War Z: zombie sci-fi horror infection

I Am Legend: horror sci-fi zombie survivalist infection

输出样例

4/5

说明：第二电影就是I Am Legend共有5个关键词，其中有4个关键词也在第一部电影里面，所以两部电影的相似程度为五分之四。

```cpp
#include<iostream>
#include<string>
#include<sstream>
#include<string>
#include<set>
using namespace std;
int main() {
    string a,b,word;
    getline(cin,a); getline(cin,b);
    a=a.substr(a.find(':')+2);
    b=b.substr(b.find(':')+2);
    set<string> x,y;
    stringstream ss;
    ss<<a;
    while(ss>>word) x.insert(word);
    ss.clear();
    ss<<b;
    while(ss>>word) y.insert(word);
    set<string>::iterator it;
    int c=0;
    for(it=y.begin();it!=y.end();it++)
        if(x.count(*it)>0) c++;
    cout<<c<<'/'<<y.size()<<endl;
    return 0;
}
```

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

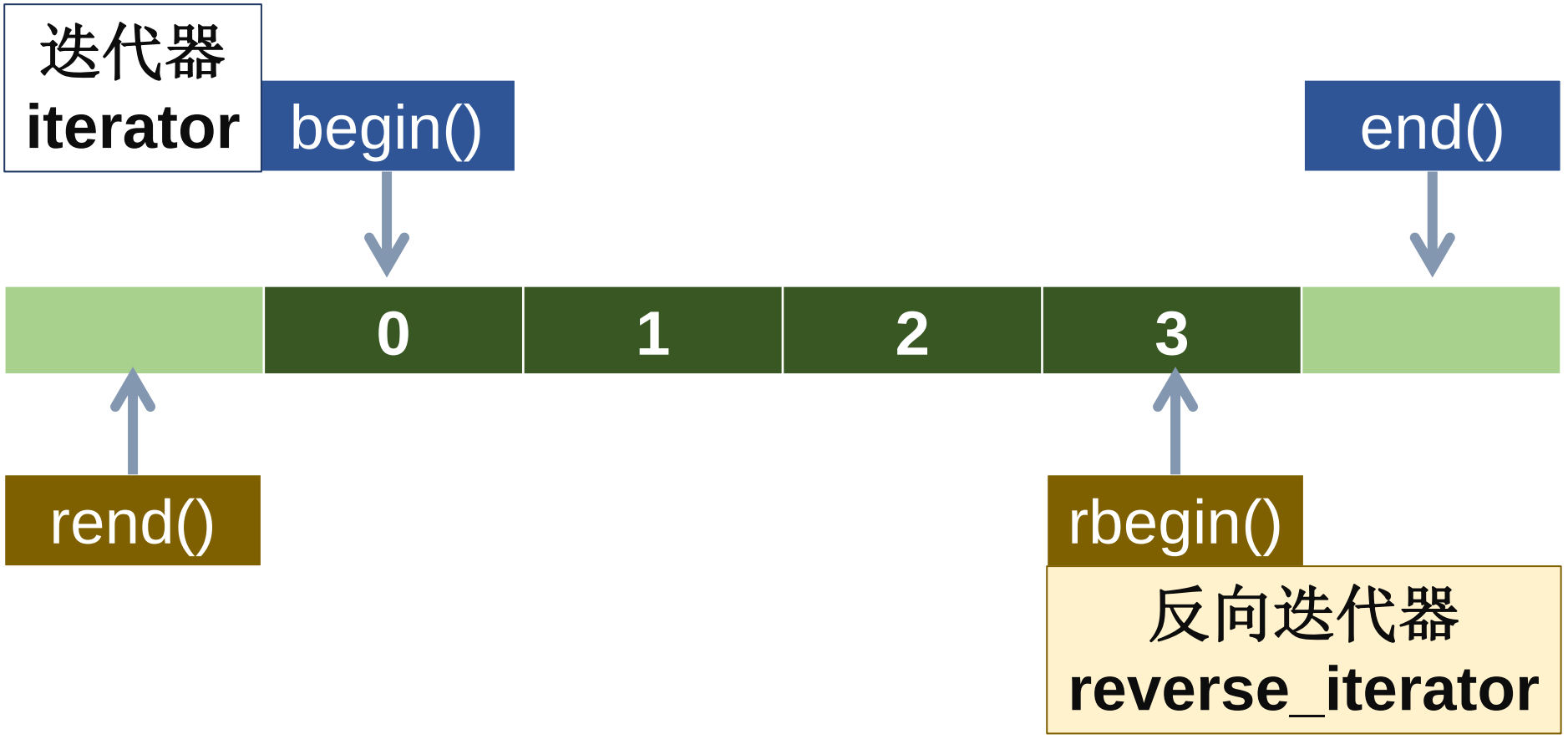
查找　计数　去重

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

例题：世界杯

世界杯小组赛如火如荼地开展，今天共有n场比赛，每场比赛i都有一个开始时间ti（以分钟为单位），如果两场比赛的开始时间相差大于等于200分钟，那么这两场球赛可以安排在一个足球场进行。否则，由于时间间隔太短容易引起骚乱，必须将两场球赛分开到不同球场。输入第一行为n，第二行为n个正整数分别代表每场比赛的开赛时间。输出至少需要几个足球场。0<=n<=16

样例输入
5
30 10 20 200 210

样例输出
4

样例输入
2
30 30

样例输出
2

例题：世界杯

定义一个multiset容器叫games
存放所有比赛的开赛时间

定义一个multiset容器叫stadiums
存放现有足球场的当前球赛的开赛时间

按照开赛时间先后，依次查看每场比赛：

如果最早空闲的球场不能举办这场比赛，
就新增一个足球场
否则就在最早空闲球场举办这场比赛

```
5       multiset<int> games,stadiums;
6       multiset<int>::iterator itg,its;
7       int n,x;     cin>>n;
8       for(int i=0;i<n;i++){
9           cin>>x; games.insert(x);
10      }
```

```
11      int ans=0;
12      for(itg=games.begin();itg!=games.end();itg++){
13          if(stadiums.empty()) stadiums.insert(*itg);
14          else {
15              its=stadiums.begin();
16              if(*itg-*its<200)
17                  stadiums.insert(*itg);
18              else {
19                  stadiums.erase(its);
20                  stadiums.insert(*itg);
21              }
22          }
23          int tot=stadiums.size();
24          ans=max(ans,tot);
25      }
```

例题：数字合并

有n个正整数，现在进行若干次操作：每次删去2个数a和b，然后加入1个数a*b+1。反复操作直到只有一个数，求最小剩下几？（1<=n<=1000）

样例输入
3
1 2 3

样例输出
8

样例输入
6
8 6 5 9 7 1

样例输出
15367

例题：数字合并

定义一个multiset容器叫s
存放所有目前的数字

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

```
5     multiset<int> s;
6     multiset<int>::iterator it;
7     int n,x; cin>>n;
8     for(int i=0;i<n;i++) {
9         cin>>x; s.insert(x);
10    }
11    while(s.size()>1){
12        it=s.end(); it--;
13        int a=*it; s.erase(it);
14        it=s.end(); it--;
15        int b=*it; s.erase(it);
16        s.insert(a*b+1);
17    }
18    cout<<*(s.begin())<<endl;
```

例题：木条切割

有一根长木条能恰好被切割成n块。这n块的长度是$L_1,L_2,L_3,...,L_n$。开始时，木条总长为$(L_1+L_2+L_3+...+L_n)$。每次切断一段木条时，费用为这段木条的长度。如果按照要求切割完成，最少需要多少费用？

$1<=n<=100$

样例输入

3

8 8 5

样例输出

34

例题：木条切割

定义一个multiset容器叫s
存放所有目前的木条长度

贪心算法选长度，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

```
5       multiset<int> s;
6       multiset<int>::iterator it;
7       int n,L,cost=0;
8       cin>>n;
9       for(int i=0;i<n;i++) {
10          cin>>L; s.insert(L);
11      }
12      while(s.size()>1){
13          it=s.begin();
14          int a=*it; s.erase(it);
15          it=s.begin();
16          int b=*it; s.erase(it);
17          s.insert(a+b);
18          cost += a+b;
19      }
20      cout<<cost<<endl;
```

参考资料

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

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

multiset 和 set 综合练习