C++编程

面条切割1

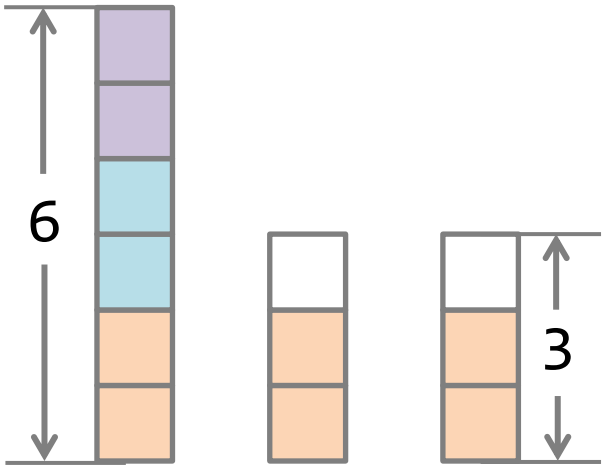
有n根面条。每根面条可以切割开成为若干正整数长度小面条，但不能拼接。现在要从这些面条中切割出长度相同的小面条，每段小面条的长度是len，允许有剩余。请问最多能切出几条小面条?

输入第一行是正整数n和len，$n<=10^5$，$len<=10^6$。第二行是n 个不超过 10^6 的正整数，表示每条的长度。

输出一个整数，代表最多能切出几条小面条。

输入样例:
3 2
6 3 3

输出样例:
5



补全程序

```
 9    cin>>n>>len;
10    for(ll i=0;i<n;i++)cin>>x[i];
11    ll cnt=0;
12    for(ll i=0;i<n;i++){
13        cnt+=
14    }
15    cout<<cnt<<endl;
```

简化版：面条切割1

有n根面条。每根面条可以切割开成为若干正整数长度小面条，但不能拼接。现在要从这些面条中切割出长度相同的小面条，每段小面条的长度是len，允许有剩余。

是非题	请问最多能切出小面条能否达到m根?

输入n,len,m已经n根面条长度。输出Yes或No

既然问题简化了 程序能否加速?	若已经完成m根切割 可以提前结束程序

补全程序

```
9     cin>>n>>len>>m;
10    for(ll i=0;i<n;i++)cin>>x[i];
11    ll cnt=0;
12    for(ll i=0;i<n;i++){
13        cnt+=x[i]/len;
14        if(          ){
15            cout<<"Yes"<<endl;
16            return 0;
17        }
18    }
19    cout<<"No"<<endl;
```

封装函数

```
6  bool OK(ll len){
7      ll cnt=0;
8      for(ll i=0;i<n;i++){
9          cnt+=x[i]/len;
10         if(cnt>=m)
11             return 1;
12     }
13     return 0;
14 }
15 int main(){
16     cin>>n>>len>>m;
17     for(ll i=0;i<n;i++)cin>>x[i];
18     if(OK(len)) cout<<"Yes"<<endl;
19     else cout<<"No"<<endl;
20     return 0;
21 }
```

OK(len)返回值代表
能否切出m条
长度为len的小面条

面条切割2

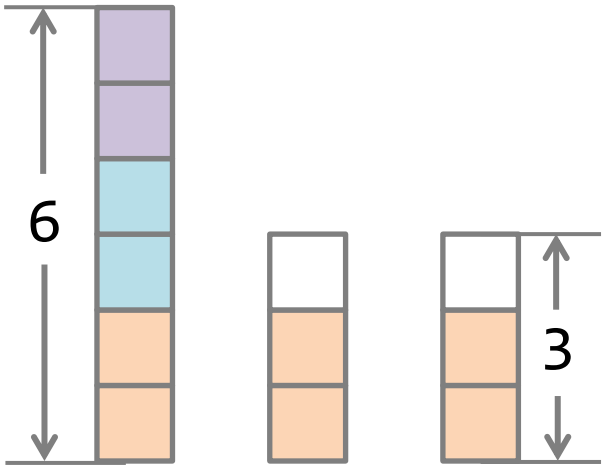
有n根面条，每根面条可以切割成为若干正整数长度小面条，但不能拼接。要从这些面条中切割出至少m条长度相同的小面条，求每段小面条的最大长度是多少。允许有剩余

输入第一行是正整数n和m，$n<=10^5$，$m<=10^8$。第二行是n 个不超过10^6的正整数，表示每条的长度。

输出切出小面条的最大长度，若无法切割，输出Failed。

输入样例：
3 5
6 3 3

输出样例：
2



有n根面条,已知第i根面条长度x[i]

面条切割1	约束	每段小面条的长度是len	目标	请问最多能切出几条小面条?
面条切割2	约束	切割出至少m条长度相同的小面条	目标	求每段小面条的最大长度是多少?

请阐述以上两题的关联性

枚举答案
不同于
枚举决策

枚举答案

再判断可行性

最优化问题的重要解法

有n根面条,已知第i根面条长度x[i]

面条切割2	约束	切割出至少m条长度相同的小面条	目标	求每段小面条的最大长度是多少?

枚举答案len	1,2,3,...一个个试
OK(len)返回值代表能否切出m条长度为len的小面条	OK(1),OK(2),OK(3)一个个判断直到发现OK(x)返回0
	答案就是x-1

补全程序

OK(len)返回值代表
能否切出m条长度为len的小面条

```
6  bool OK(ll len){
7      ll cnt=0;
8      for(ll i=0;i<n;i++){
9          cnt+=x[i]/len;
10         if(cnt>=m) return ;
11     }
12     return ;
13 }
```

OK(len) 时间复杂度是多少?	O(n)

手算样例

输入
n=4,m=6
四根长度 5 6 7 8

OK(len)返回值代表
能否切出m条长度为len的小面条

len	1	2	3	4	5	6	7	8
OK(len)	1	1	1	0	0	0	0	0

答案	每段小面条 最大长度

补全程序

```
19      ll ans=0;
20      ll l=1;
21      ll r=*max_element(x,x+n);
22      for(ll len=1;len<=r;len++)
23          if(          ){
24              ans=len-1;
25              break;
26          }
27      if(ans==0)cout<<"Failed"<<endl;
28      else cout<<ans<<endl;
```

r代表最大面条长度

r代表可能的最大答案

这个程序时间复杂度是多少?

O(nr)

能否再加快速度?

有n根面条,已知第i根面条长度x[i]

面条 切割2	约束	切割出至少m条长度相同的小面条	目标	求每段小面条的最大长度是多少?

二分枚举答案len	若OK(x)返回1就尝试更长的长度
	若OK(x)返回0就尝试更短的长度
OK(len)返回值代表 能否切出m条 长度为len的小面条	

手算样例

输入
n=4,m=6
四根长度 5 6 7 8

l代表答案可能范围的左端点
r代表答案可能范围的右端点
ans代表当前最优答案

初始化	l=1	取左右端点的中间点	
	r=8	mid=4	
	ans=0	OK(mid)返回几?	0
更新	l=1	取左右端点的中间点	
	r=3	mid=2	
	ans=0	OK(mid)返回几?	1
更新	l=3	取左右端点的中间点	
	r=3	mid=3	
	ans=2	OK(mid)返回几?	1
更新	l=4		
	r=3		
	ans=3		

补全程序

OK(len)返回值代表
能否切出m条长度为len的小面条

```
6  bool OK(ll len){
7      ll cnt=  ;
8      for(ll i=0;i<n;i++){
9          cnt+=
10         if(        ) return 1;
11     }
12     return 0;
13 }
```

OK(len) 时间复杂度是多少?	O(n)

l代表答案可能范围的左端点
r代表答案可能范围的右端点
ans代表当前最优答案

```
19      ll l=1;
20      ll r=*max_element(x,x+n);
21      ll ans=0;
22      while(l<=r){
23          ll mid=l+(r-l)/2;
24          if(OK(mid))ans=mid,l=mid+1;
25          else r=mid-1;
26      }
```

这个程序 时间复杂度是多少?	O(nlogn)

补全程序

l代表答案可能范围的左端点
r代表答案可能范围的右端点
ans代表当前最优答案

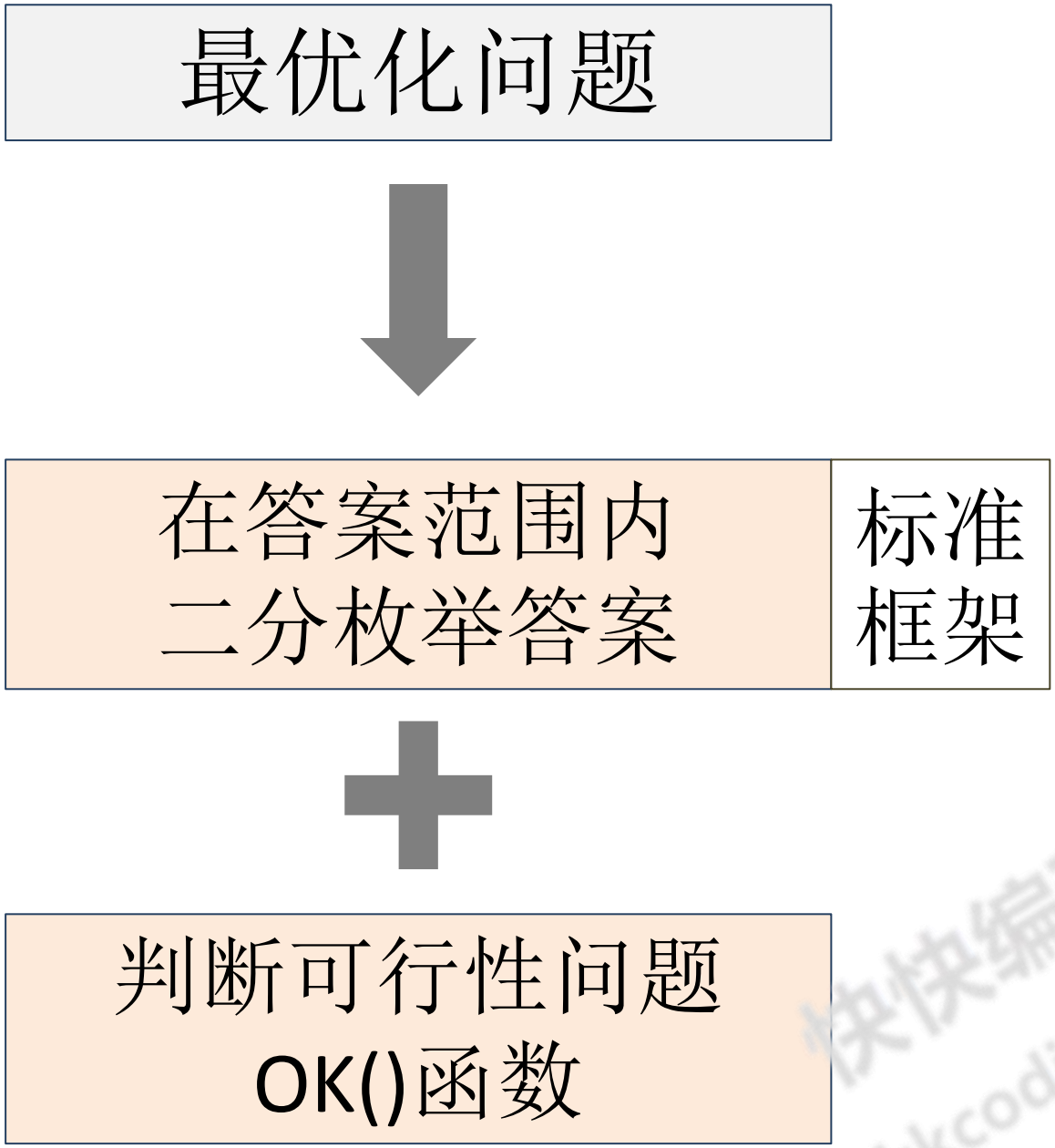
```
19    ll l=
20    ll r=
21    ll ans=
22    while(l<=r){
23        ll mid=l+(r-l)/2;
24        if(OK(mid))ans=mid,l=mid+1;
25        else r=mid-1;
26    }
```

补全程序

l代表答案可能范围的左端点
r代表答案可能范围的右端点
ans代表当前最优答案

```
19    ll l=1;
20    ll r=*max_element(x,x+n);
21    ll ans=0;
22    while(      ){
23        ll mid=l+(r-l)/2;
24        if(OK(mid))
25        else
26    }
```

只有OK了 才更新答案
不OK时 降低要求



最大化问题：二分整数答案框架

```
int l=
int r=
int ans=
while(l<=r){
        int mid=l+(r-l)/2;
        if(OK(mid))
        else
}
```

初始化左端点，即答案可能的最小值

初始化右端点，即答案可能的最大值

初始化答案尽量小

当还存在待查找范围

二分范围：中点为mid

ans=mid, l=mid+1;

r=mid-1;

最小化问题：二分整数答案框架

```
int l=
int r=
int ans=
while(l<=r){
        int mid=l+(r-l)/2;
        if(OK(mid))
        else
}
```

初始化左端点，即答案可能的最小值

初始化右端点，即答案可能的最大值

初始化答案尽量大

当还存在待查找范围

二分范围：中点为mid

ans=mid,r=mid-1;

l=mid+1;

最大化问题	初始化答案要小	逐步变大

最小化问题	初始化答案要大	逐步变小

臭皮匠1

共n个臭皮匠坐成一排，从左数第i个人智商为xi，他们想去分组挑战聪明的诸葛亮。每组只可以安排相邻就坐的若干个臭皮匠上场挑战。为了让每组都有资格挑战诸葛亮，必须保证每一组的智商和不低于m，**求最多派出几组臭皮匠?**

输入第一行为正整数n和m，第二行为n个正整数xi。n<=100，m<=1000,xi<=500，输出一个整数。

输入样例: 5 100 50 100 60 60 110
输出样例: 3

输入样例: 5 100 20 10 10 30 29
输出样例: 0

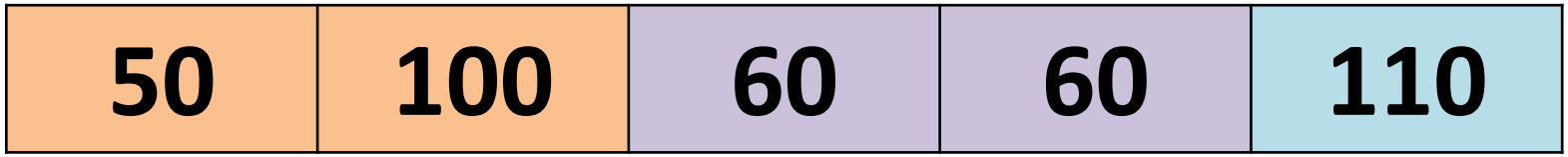
贪心法

输入样例：
5 100
50 100 60 60 110

输出样例：
3

从左向右每次安排
1个臭皮匠加入当前小组

当前小组如果凑满m的智商
就新开一组



```cpp
#include<iostream>
using namespace std;
const int N=109;
int n,m,x[N];
int main(){
    cin>>n>>m;
    for(int i=0;i<n;i++) cin>>x[i];
    int cnt=0,sum=0;
    for(int i=0;i<n;i++){
        sum+=x[i];
        if(sum>=m){
            sum=0;
            cnt++;
        }
    }
    cout<<cnt<<endl;
    return 0;
}
```

cnt代表当前已有几个小组完成智商和不低于m的要求

sum记录当前该小组成员智商和

臭皮匠2

共n个臭皮匠坐成一排，第i个人智商为xi，他们想去分组挑战聪明的诸葛亮。每一组只可以安排相邻就坐的若干个臭皮匠上场挑战。现在必须组织起k次挑战，为了不要让围观者看不起臭皮匠团体，**希望使各组臭皮匠智商和的最小值越大越好，请问这个数最大是多少？**

输入第一行为正整数n和k，第二行为n个正整数xi。1<=k<=n<=100，xi<=500，输出一个正整数。

输入样例： 5 3 50 100 60 60 110
输出样例： 110

输入样例： 3 2 20 20 30
输出样例： 30

和臭皮匠1有啥关系

臭皮匠2

臭皮匠2算法：暴力枚举答案

从小到大枚举可能的答案: m=0,1,2,3,...,50000

对于特定的m,利用臭皮匠1的算法判断:**保证每一组的智商和不低于m时，能否派出k组臭皮匠？**

如果发现对于某个m无法派出k组臭皮匠，就停止枚举，输出答案m-1

时间复杂度O(M*N)
M是答案可能的范围大小
N是臭皮匠人数的上限

臭皮匠2

臭皮匠2算法：二分枚举答案

二分枚举可能的答案: m 范围初始化为[0,50000]中整数

对于特定的m,利用臭皮匠1的算法判断:**保证每一组的智商和不低于m时，能否派出k组臭皮匠？**

不断二分缩小m的可能范围，直到待查找范围为空

时间复杂度O(N*logM)
M是答案可能的范围大小
N是臭皮匠人数的上限

小结

最小值最大化问题

最大值最小化问题

常用方法
二分答案+判断可行性

记笔记

OK(m)判断:
每组智商和不低于m时
能否派出k组

```
6  bool OK(int m){
7      int cnt=0,sum=0;
8      for(int i=0;i<n;i++)
9          if((sum+=x[i])>=m)sum=0,cnt++;
10     return cnt>=k;
11 }
```

```
15     int l=*min_element(x,x+n);
16     int r=0;
17     for(int i=0;i<n;i++)r+=x[i];
18     int ans=l;
19     while(l<=r){
20         int mid=l+(r-l)/2;
21         if(OK(mid)) ans=mid,l=mid+1;
22         else r=mid-1;
23     }
24     cout<<ans<<endl;
```

初始化左端点:最低智商

初始化右端点:智商总和

初始化答案

当还存在待查找范围

二分范围，中点为mid

若mid是可行解，就找更大可行解；
否则，就找更小可行解

易错点

老师打开程序
“整数二分查找易错点汇总”

针对"臭皮匠2"问题

观察2分钟后
找出程序内所有错误

现场挑战
快快编程436

```
6     cin>>n>>m;
7     for(int i=0;i<n;i++) cin>>x[i];

8     int cnt=1;
9     int sum=0;

10    int i;
11    for(i=0;i<n;i++){
12        if(x[i]>m)
13        sum+=x[i];
14        if(sum>m){
15            
16            cnt++;
17        }
18    }

19    if(i<n)cout<<0<<endl;
20    else cout<<cnt<<endl;
```

cnt表示当前用了几辆车

sum表示当前车内智商和

补全程序

现场挑战
快快编程437

437算法分析

二分枚举答案+判断可行性

OK(x)代表什么含义?

记笔记

OK(x)判断:
每组智商和不超过x时
能否将n人分组控制在k组以内

建议参考完成436

OK(x)判断：
每组智商和不超过x时
能否将n人分组控制在k组以内

```
5  bool OK(int m){
6      int cnt=  ,sum=0;
7      for(int i=0;i<n;i++){
8          if(x[i]>m)return 0;
9          sum+=x[i];
10         if(          ){
11             cnt++;
12             sum=x[i];
13         }
14     }
15     return           
16 }
```

OK(x)判断：
每组智商和不超过x时
能否将n人分组控制在k组以内

```
20      cin>>n>>k;
21      for(int i=0;i<n;i++)cin>>x[i];
22      int l=0;
23      int r=0;
24      for(int i=0;i<n;i++)r+=x[i];
25      
26      while(      ){
27          int mid=l+(r-l)/2;
28          if(OK(mid))
29          else
30      }
31      cout<<ans<<endl;
```

快快编程作业

654

436

437

建议参考436

拓展题	438(参考997),1742,440