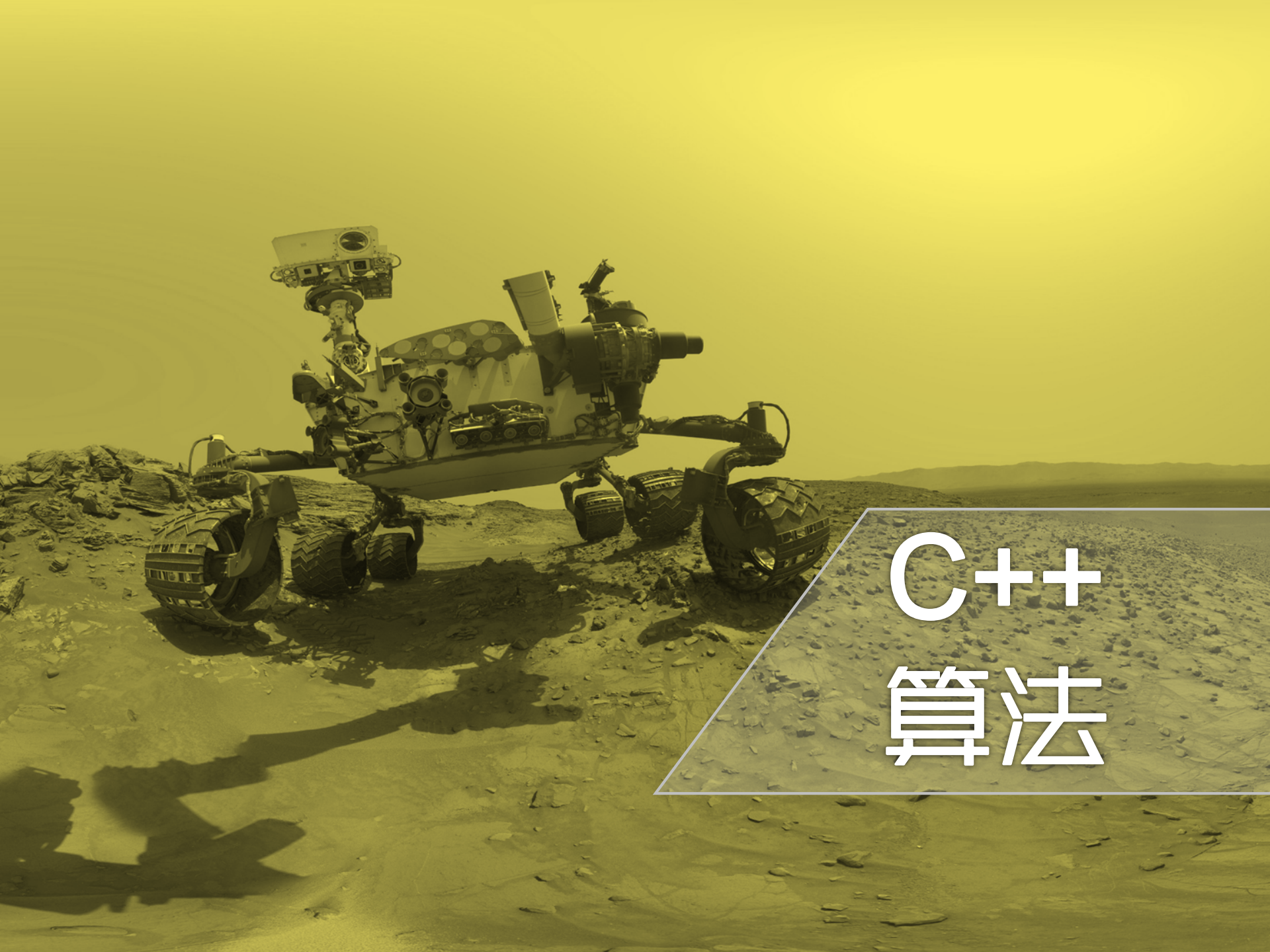

C++
算法

单调子序列问题

Monotonic Subsequences

各种子序列(标记红色)

上升子序列

不升子序列

下降子序列

下降子序列

单调子序列问题

最长上升子序列

最长不升子序列

最长下降子序列

最长不降子序列

上升子序列最小划分数

不升子序列最小划分数

下降子序列最小划分数

不降子序列最小划分数

单调子序列问题

最长上升子序列

8个问题的核心 其实是2个问题	本质是同 1个算法

不升子序列最小划分数

例题：最长上升子序列

The **longest increasing subsequence (LIS)** problem is to find a subsequence of a given sequence in which the subsequence's elements are in sorted order, highest to lowest, and in which the subsequence is as long as possible.

输入样例
8
2 1 2 7 2 3 8 4

输入第一行为n代表序列长度。
输入第二行为序列的n个数字:x[0],x[1],...,x[n-1]
输出上升子序列最长的长度。

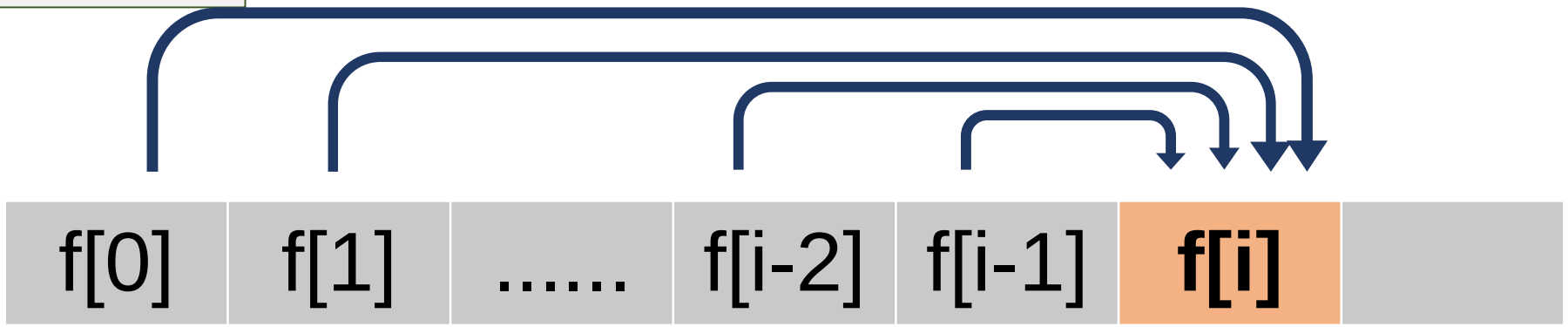
输出样例
4

思考：如何设计状态?
如何设计f[i]代表什么含义?
如何建立f[i]和f[i-1],f[i-2],...的联系?

LIS：解法1

$f[i]$代表以$x[i]$结尾的上升子序列最长能有多长

状态i描述子序列结尾的位置



计算f[i]时参考
f[0], f[1], . . . , f[i−1]的结果

LIS：解法1

$f[i]$代表以$x[i]$结尾的上升子序列最长能有多长

状态i描述子序列结尾的位置

当i为0时	$f[0] = 1$

初始条件

当i>=1时	$f[i] = max_{0 \le j \le i-1}\{f[j] \mid x[j] < x[i]\} + 1$

状态转移方程

计算f[i]时参考
f[0], f[1], . . . , f[i−1]的结果

$ans = max_i\{f[i]\}$

最终答案

复杂度O(N^2)，能否更快？

LIS：解法1

```
1  #include<iostream>
2  #include<algorithm>
3  #define N 1005
4  using namespace std;
5  int n,f[N],x[N];
6  int main(){
7      cin>>n;
8      for(int i=0;i<n;i++)cin>>x[i];
9      f[0]=1;
10     for(int i=1;i<n;i++){
11         f[i]=1;
12         for(int j=0;j<i;j++)
13             if(x[j]<x[i])f[i]=max(f[i],f[j]+1);
14     }
15     cout<<*max_element(f,f+n)<<endl;
16     return 0;
17 }
```

例题：不升子序列最小划分数

原序列共有n个整数数字，需要将原序列划分，
组成几条不上升子序列，求最少分成几条。
注意1：这几条不升子序列需要涵盖所有n个数字
注意2：n个数字需要分别归属于不同的子序列
输入第一行为n代表序列长度。n<=1000
输入第二行为序列的n个数字:x[0],x[1],...,x[n-1]
输出不升子序列最小划分数

输入样例
6
2 1 2 7 2 3

输出样例
3

样例最少3条
不升子序列
例如：
2 1
2 2
7 3

i	x[i]
0	2
1	1
2	2
3	7
4	2
5	3

贪心法：不升子序列最小划分数

贪心算法：

依次循环安排每个数字x[i]：

1.如果x[i]无法接在已有子序列之后，

就新增加一个子序列，安排x[i]

2.如果x[i]可以接在已有子序列后，

就在可选子序列中挑选最小数的结尾后，安排x[i]

d[i]代表第i条不升子序列的最后一个数字

贪心法步骤：依次根据x[i]不断修改d数组

贪心法：不升子序列最小划分数

d[i]代表第i条不升子序列的最后一个数字

i	x[i]
0	2
1	1

d[0]	d[1]	d[2]	d[3]	d[4]	d[5]	
∞	∞	∞	∞	∞	∞	
2	∞	∞	∞	∞	∞	1条不升子序列
1	∞	∞	∞	∞	∞	1条不升子序列

贪心法：不升子序列最小划分数

d[i]代表第i条不升子序列的最后一个数字

i	x[i]
0	2
1	1
2	2

d[0]	d[1]	d[2]	d[3]	d[4]	d[5]	
∞	∞	∞	∞	∞	∞	
2	∞	∞	∞	∞	∞	1条不升子序列
1	∞	∞	∞	∞	∞	1条不升子序列
1	2	∞	∞	∞	∞	2条不升子序列

贪心法：不升子序列最小划分数

d[i]代表第i条不升子序列的最后一个数字

i	x[i]
0	2
1	1
2	2
3	7

d[0]	d[1]	d[2]	d[3]	d[4]	d[5]	
∞	∞	∞	∞	∞	∞	
2	∞	∞	∞	∞	∞	1条不升子序列
1	∞	∞	∞	∞	∞	1条不升子序列
1	2	∞	∞	∞	∞	2条不升子序列
1	2	7	∞	∞	∞	3条不升子序列

贪心法：不升子序列最小划分数

d[i]代表第i条不升子序列的最后一个数字

i	x[i]
0	2
1	1
2	2
3	7
4	2

d[0]	d[1]	d[2]	d[3]	d[4]	d[5]	
∞	∞	∞	∞	∞	∞	
2	∞	∞	∞	∞	∞	1条不升子序列
1	∞	∞	∞	∞	∞	1条不升子序列
1	2	∞	∞	∞	∞	2条不升子序列
1	2	7	∞	∞	∞	3条不升子序列
1	2	7	∞	∞	∞	3条不升子序列

贪心法：不升子序列最小划分数

d[i]代表第i条不升子序列的最后一个数字

i	x[i]
0	2
1	1
2	2
3	7
4	2
5	3

d[0]	d[1]	d[2]	d[3]	d[4]	d[5]	
∞	∞	∞	∞	∞	∞	
2	∞	∞	∞	∞	∞	1条不升子序列
1	∞	∞	∞	∞	∞	1条不升子序列
1	2	∞	∞	∞	∞	2条不升子序列
1	2	7	∞	∞	∞	3条不升子序列
1	2	7	∞	∞	∞	3条不升子序列
1	2	3	∞	∞	∞	3条不升子序列

贪心法：代码1

```
#include<iostream>
#include<algorithm>
#define N 1005
using namespace std;
int n,i,j,d[N],x[N];
int main(){
    cin>>n;
    for(int i=0;i<n;i++)cin>>x[i];
    int cnt=0;
    for(i=0;i<n;i++){
        for(j=0;j<cnt;j++)
            if(d[j]>=x[i])break;
        d[j]=x[i];
        if(j==cnt)cnt++;
    }
    cout<<cnt<<endl;
    return 0;
}
```

复杂度$O(N^2)$
能否更快？

贪心法：代码2

d[i]代表i号不升子序列的最后一个数字

思考：为什么d数组随时保持着从小到大排序?

```
1  #include<iostream>
2  #include<algorithm>
3  #define N 1005
4  #define INF 2e9
5  using namespace std;
6  int n,d[N],x[N];
7  int main(){
8      cin>>n;
9      for(int i=0;i<n;i++)cin>>x[i];
10     fill(d,d+n,INF);
11     for(int i=0;i<n;i++)
12         *lower_bound(d,d+n,x[i])=x[i];
13     int cnt=lower_bound(d,d+n,INF)-d;
14     cout<<cnt<<endl;
15     return 0;
16 }
```

复杂度
O(NlogN)

Dilworth定理

链的最长长度 = 反链划分数最小值

Dilworth定理

链的最长长度 = 反链划分数最小值

链：上升子序列　　反链：不升子序列

链：不降子序列　　反链：下降子序列

Dilworth定理

最长上升子序列	不升子序列最小划分数

最长的长度为LIS = 最小划分数为M

1.证明LIS≥M

在贪心法求解“不升子序列最小划分数M”时构造出了d数组记录每一条不升子序列的尾数。最终d数组形成一条上升子序列，该长度不可能超过LIS。

2.证明LIS≤M

对于一条给定的上升子序列L，L其中任意2个元素都不可能在同一条不升子序列中出现。L的每个元素都分别在不同的不升子序列中。

LIS解法2：不升子序列最小划分数

```
1  #include<iostream>
2  #include<algorithm>
3  #define N 1005
4  #define INF 2e9
5  using namespace std;
6  int n,d[N],x[N];
7  int main(){
8      cin>>n;
9      for(int i=0;i<n;i++)cin>>x[i];
10     fill(d,d+n,INF);
11     for(int i=0;i<n;i++)
12         *lower_bound(d,d+n,x[i])=x[i];
13     int cnt=lower_bound(d,d+n,INF)-d;
14     cout<<cnt<<endl;
15     return 0;
16 }
```

现场挑战

请复习LIS的O(NlogN)算法代码
闭卷用纸和笔
写出LIS问题的完整程序

现场限时8分钟

请附近同学互相找bug

单调子序列问题综合练习

上升子序列最小划分数

原序列共有n个整数数字，需要将原序列划分，
组成几条上升子序列，求最少分成几条。
注意1：这几条不升子序列需要涵盖所有n个数字
注意2：n个数字需要分别归属于不同的子序列
输入第一行为n代表序列长度。
输入第二行为序列的n个数字:x[0],x[1],...,x[n-1]
输出上升子序列最小划分数

输入样例
6
2 1 2 7 2 3

输出样例
3

样例至少3条上升子序列
例如:
2 7
1 2 3
2

i	x[i]
0	2
1	1
2	2
3	7
4	2
5	3

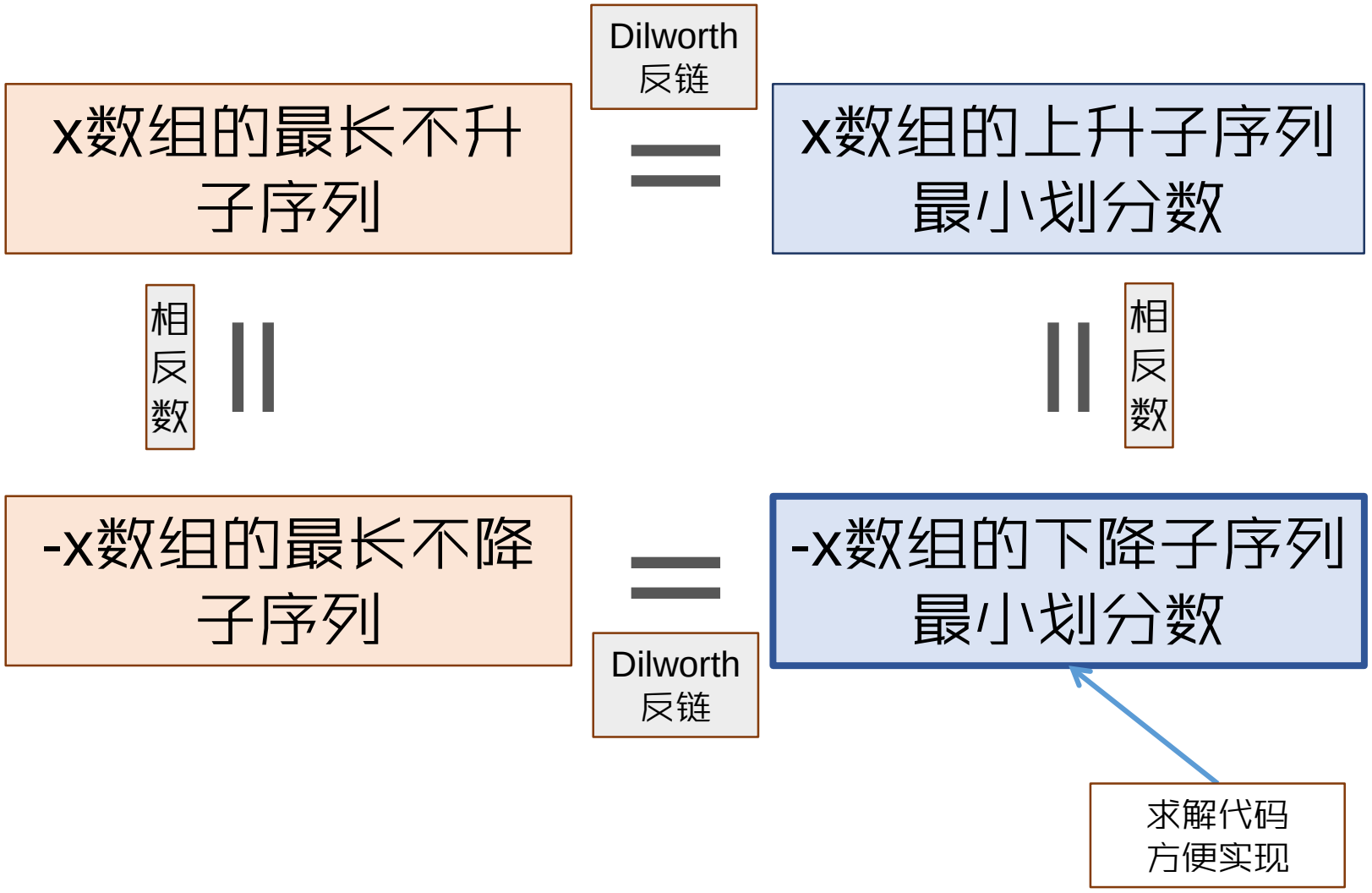
上升子序列最小划分数

x数组的上升子序列最小划分数 = -x数组的下降子序列最小划分数

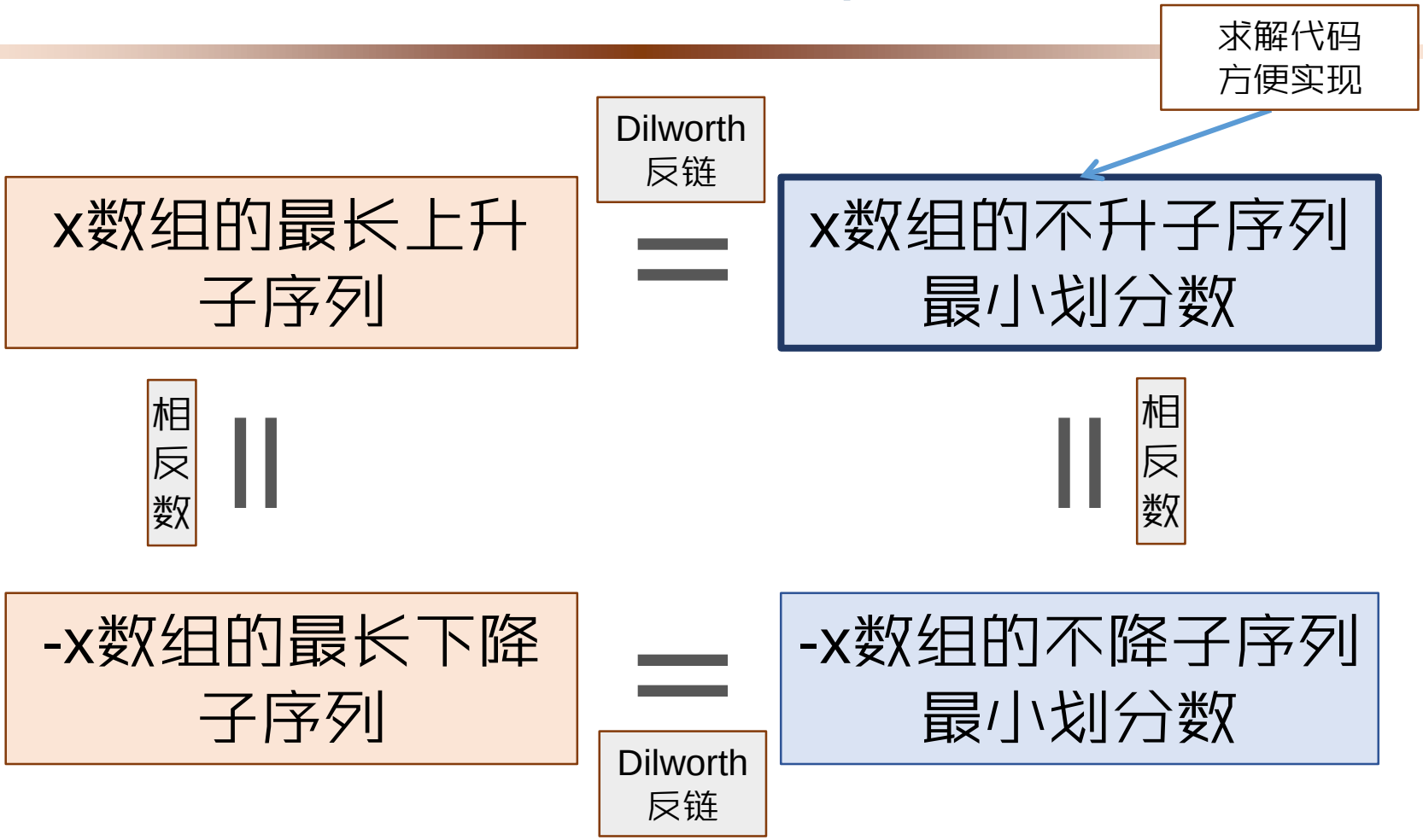
相反数

```
10      fill(d,d+n,INF);
11      for(int i=0;i<n;i++)
12          *upper_bound(d,d+n,-x[i])=-x[i];
13      int cnt=lower_bound(d,d+n,INF)-d;
14      cout<<cnt<<endl;
```

单调子序列问题


x数组的最长不升子序列
Dilworth
反链
x数组的上升子序列最小划分数
相反数
相反数
-x数组的最长不降子序列
Dilworth
反链
-x数组的下降子序列最小划分数
求解代码方便实现

单调子序列问题


求解代码
方便实现
Dilworth
反链
x数组的最长上升
子序列
=
x数组的不升子序列
最小划分数
相
反
数
||
||
相
反
数
-x数组的最长下降
子序列
=
-x数组的不降子序列
最小划分数
Dilworth
反链

作业

要求：每一题先用注释写出
该问题涉及8个问题中的哪个？

	455
	187
	189
拓展题	188