

Contents

1	ubuntu	1
1.1	run	1
1.2	cp.sh	1
2	Basic	1
2.1	ascii	1
2.2	limits	1
3	字串	2
3.1	最長迴文子字串	2
3.2	stringstream	2
4	STL	2
4.1	priority_queue	2
4.2	queue	2
4.3	deque	2
4.4	map	3
4.5	unordered_map	3
4.6	set	3
4.7	multiset	4
4.8	unordered_set	4
4.9	單調隊列	4
5	sort	5
5.1	大數排序	5
5.2	bubble sort	5
6	math	6
6.1	質數與因數	6
6.2	prime factorization	6
6.3	快速幂	6
6.4	歐拉函數	7
7	algorithm	7
7.1	basic	7
7.2	binarysearch	7
7.3	prefix sum	7
7.4	差分	7
7.5	greedy	8
8	動態規劃	11
8.1	LCS 和 LIS	11
9	graph	11
9.1	graph	11
10	Section2	12
10.1	thm	12
11	space	12
11.1	s	12
12	reference	13
12.1	assert.cpp	13
12.2	print.cpp	13
12.3	rvec.cpp	13
12.4	find.cpp	13
12.5	sort.cpp	13
12.6	minmax.cpp	13
12.7	hw6.cpp	14
12.8	rectangle.cpp	15
12.9	SortFunctionTemplate.cpp	15
12.10	exception.cpp	16
12.11	RationalNumber.cpp	16
12.12	purevirtual.cpp	17

1 ubuntu

1.1 run

1| ~\$ bash cp.sh PA

1.2 cp.sh

```
1 #!/bin/bash
2 clear
3 g++ $1.cpp -DDBG -o $1
4 if [[ "$?" == "0" ]]; then
5     echo Running
6     ./$1 < $1.in > $1.out
7     echo END
8 fi
```

2 Basic

2.1 ascii

int	char	int	char	int	char
32		64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(72	H	104	h
41)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[123	{
60	<	92	\	124	
61	=	93]	125	}
62	>	94	^	126	~
63	?	95	_		

2.2 limits

[Type]	[size]	[range]
char	1	127 to -128
signed char	1	127 to -128
unsigned char	1	0 to 255
short	2	32767 to -32768
int	4	2147483647 to -2147483648
unsigned int	4	0 to 4294967295
long	4	2147483647 to -2147483648
unsigned long	4	0 to 18446744073709551615
long long	8	
	9223372036854775807 to -9223372036854775808	
double	8	1.79769e+308 to 2.22507e-308
long double	16	1.18973e+4932 to 3.3621e-4932
float	4	3.40282e+38 to 1.17549e-38
unsigned long long	8	0 to 18446744073709551615
string	32	

3 字串

3.1 最長迴文子字串

```

1 #include <bits/stdc++.h>
2 #define T(x) ((x) % 2 ? s[(x) / 2] : '. ')
3 using namespace std;
4
5 string s;
6 int n;
7
8 int ex(int l, int r) {
9     int i = 0;
10    while(l - i >= 0 && r + i < n && T(l - i) == T(r + i)) i++;
11    return i;
12 }
13
14 int main() {
15     cin >> s;
16     n = 2 * s.size() + 1;
17
18     int mx = 0;
19     int center = 0;
20     vector<int> r(n);
21     int ans = 1;
22     r[0] = 1;
23     for(int i = 1; i < n; i++) {
24         int ii = center - (i - center);
25         int len = mx - i + 1;
26         if(i > mx) {
27             r[i] = ex(i, i);
28             center = i;
29             mx = i + r[i] - 1;
30         } else if(r[ii] == len) {
31             r[i] = len + ex(i - len, i + len);
32             center = i;
33             mx = i + r[i] - 1;
34         } else {
35             r[i] = min(r[ii], len);
36         }
37         ans = max(ans, r[i]);
38     }
39
40     cout << ans - 1 << "\n";
41     return 0;
42 }

```

3.2 stringstream

```

1 string s, word;
2 stringstream ss;
3 getline(cin, s);
4 ss << s;
5 while(ss >> word)
6     cout << word << endl;

```

4 STL

4.1 priority_queue

```

1 priority_queue: 優先隊列，資料預設由大到小排序，
   即優先權高的資料會先被取出。
2 宣告：
3     priority_queue<int> pq;
4 把元素 x 加進 priority_queue：
5     pq.push(x);
6 讀取優先權最高的值：
7     x = pq.top();
8     pq.pop();           //讀取後刪除

```

```

9 判斷是否為空的priority_queue：
10    pq.empty()           //回傳 true
11    pq.size()            //回傳 0
12 如需改變priority_queue的優先權定義：
13    priority_queue<T> pq; //預設由大到小
14    priority_queue<T, vector<T>, greater<T>> > pq;
15                                //改成由小到大
16    priority_queue<T, vector<T>, cmp> pq; //cmp

```

4.2 queue

```

1 queue: 佇列，資料有「先進先出」(first in first out,
   FIFO)的特性。
2 就像排隊買票一樣，先排隊的客戶被服務。
3 宣告：
4     queue<int> q;
5 把元素 x 加進 queue：
6     q.push(x);
7 取值：
8     x = q.front(); //頭
9     x = q.back();  //尾
10 移除已經讀取的值：
11     q.pop();
12 判斷是否為空的queue：
13     q.empty() 回傳 true
14     q.size()  回傳零
15
16 #include <iostream>
17 #include <queue>
18 using namespace std;
19
20 int main() {
21     int n;
22     while (cin >> n) {
23         if (n == 0) break;
24         queue<int> q;
25         for (int i = 0; i < n; i++) {
26             q.push(i+1);
27         }
28         cout << "Discarded cards:";
29         for (int i = 0; i < n-1; i++) {
30             if (i != 0) cout << ', ';
31             cout << ' ' << q.front();
32             q.pop();
33             q.push(q.front());
34             q.pop();
35         }
36         cout << endl << "Remaining card: " <<
37             q.front() << endl;
38     }
39 }

```

4.3 deque

```

1 deque 是 C++ 標準模板函式庫
2 (Standard Template Library, STL)
3 中的雙向佇列容器 (Double-ended Queue)，
4 跟 vector 相似，不過在 vector
5 中若是要添加新元素至開端，
6 其時間複雜度為 O(N)，但在 deque 中則是 O(1)。
7 同樣也能在我們需要儲存更多元素的時候自動擴展空間，
   讓我們不必煩惱佇列長度的問題。
8 dq.push_back() //在 deque 的最尾端新增元素
9 dq.push_front() //在 deque 的開頭新增元素
10 dq.pop_back()   //移除 deque 最尾端的元素
11 dq.pop_front()  //移除 deque 最開頭的元素
12 dq.back()       //取出 deque 最尾端的元素
13 dq.front()      //回傳 deque 最開頭的元素
14 dq.insert()
15 dq.insert(position, n, val)

```

```

16 position: 插入元素的 index 值
17 n: 元素插入次數
18 val: 插入的元素值
19 dq.erase()
    //刪除元素，需要使用迭代器指定刪除的元素或位置，
    同時也會返回指向刪除元素下一元素的迭代器。
20 dq.clear()    //清空整個 deque 佇列。
21 dq.size()    //檢查 deque 的尺寸
22 dq.empty()   //如果 deque 佇列為空返回 1；
                若是存在任何元素，則返回0
23 dq.begin()   //返回一個指向 deque 開頭的迭代器
24 dq.end()     //指向 deque 結尾，
25             不是最後一個元素，
26             而是最後一個元素的下一個位置
27

```

4.4 map

```

1 map: 存放 key-value pairs 的映射資料結構，
2     會按 key 由小到大排序。
3 元素存取
4 operator[]: 存取指定的[i]元素的資料
5
6 迭代器
7 begin(): 回傳指向map頭部元素的迭代器
8 end(): 回傳指向map末尾的迭代器
9 rbegin(): 回傳一個指向map尾部的反向迭代器
10 rend(): 回傳一個指向map頭部的反向迭代器
11

```

遍歷整個map時，利用iterator操作：

```

13 取key: it->first 或 (*it).first
14 取value: it->second 或 (*it).second
15

```

容量

```

17 empty(): 檢查容器是否為空，空則回傳true
18 size(): 回傳元素數量
19 max_size(): 回傳可以容納的最大元素個數
20

```

修改器

```

22 clear(): 刪除所有元素
23 insert(): 插入元素
24 erase(): 刪除一個元素
25 swap(): 交換兩個map
26

```

查找

```

28 count(): 回傳指定元素出現的次數
29 find(): 查找一個元素
30

```

//實作範例

```

32 #include <bits/stdc++.h>
33 using namespace std;
34
35 int main(){
36
37     //declaration container and iterator
38     map<string, string> mp;
39     map<string, string>::iterator iter;
40     map<string, string>::reverse_iterator iter_r;
41
42     //insert element
43     mp.insert(pair<string, string>("r000",
44                                     "student_zero"));
45
46     mp["r123"] = "student_first";
47     mp["r456"] = "student_second";
48
49     //traversal
50     for(iter = mp.begin(); iter != mp.end(); iter++)
51         cout<<iter->first<<" "<<iter->second<<endl;
52     for(iter_r = mp.rbegin(); iter_r != mp.rend();
53         iter_r++)
54         cout<<iter_r->first<<" "
55             <<iter_r->second<<endl;
56

```

```

53
54     //find and erase the element
55     iter = mp.find("r123");
56     mp.erase(iter);
57
58     iter = mp.find("r123");
59
60     if(iter != mp.end())
61         cout<<"Find, the value is "
62             <<iter->second<<endl;
63     else
64         cout<<"Do not Find"<<endl;
65
66     return 0;
67 }
68
69 //map統計數字
70 #include<bits/stdc++.h>
71 using namespace std;
72
73 int main(){
74     ios::sync_with_stdio(0),cin.tie(0);
75     long long n,x;
76     cin>>n;
77     map <int,int> mp;
78     while(n--){
79         cin>>x;
80         ++mp[x];
81     }
82     for(auto i:mp) cout<<i.first<<" "<<i.second<<endl;
83 }

```

4.5 unordered_map

```

1 unordered_map: 存放 key-value pairs
2               的「無序」映射資料結構。
3 用法與map相同

```

4.6 set

1 set: 集合，去除重複的元素，資料由小到大排序。

2 宣告：

```
4 set <int> st;
```

6 把元素 x 加進 set：

```
7 st.insert(x);
```

9 檢查元素 x 是否存在 set 中：

```
10 st.count(x);
```

12 刪除元素 x：

```
13 st.erase(x); // 可傳入值或iterator
```

15 清空集合中的所有元素：

```
16 st.clear();
```

18 取值：使用iterator

```
19 x = *st.begin();
```

20 // set中的第一個元素(最小的元素)。

```
21 x = *st.rbegin();
```

22 // set中的最後一個元素(最大的元素)。

24 判斷是否為空的set：

```
25 st.empty() 回傳true
```

```
26 st.size() 回傳零
```

28 常用來搭配的member function：

```
29 st.count(x);
```

```
30 auto it = st.find(x);
```

```
31 // binary search, O(log(N))
```

```
32 auto it = st.lower_bound(x);
```

```

33 // binary search, O(log(N))
34 auto it = st.upper_bound(x);
35 // binary search, O(log(N))

```

4.7 multiset

1 與 set 用法雷同，但會保留重複的元素。
 2 資料由小到大排序。
 3 宣告：
 4 multiset<int> st;
 5 刪除資料：
 6 st.erase(val); 會刪除所有值為 val 的元素。
 7 st.erase(st.find(val)); 只刪除第一個值為 val
 的元素。

4.8 unordered_set

1 unordered_set 的實作方式通常是用雜湊表(hash table)，
 2 資料插入和查詢的時間複雜度很低，為常數級別 $O(1)$ ，
 3 相對的代價是消耗較多的記憶體，空間複雜度較高，
 4 無自動排序功能。

```

6 初始化
7 unordered_set<int> myunordered_set{1, 2, 3, 4, 5};
8
9 陣列初始化
10 int arr[] = {1, 2, 3, 4, 5};
11 unordered_set<int> myunordered_set(arr, arr+5);
12
13 插入元素
14 unordered_set<int> myunordered_set;
15 myunordered_set.insert(1);
16
17 迴圈遍歷 unordered_set 容器
18 #include <iostream>
19 #include <unordered_set>
20 using namespace std;
21
22 int main() {
23     unordered_set<int> myunordered_set = {3, 1};
24     myunordered_set.insert(2);
25     myunordered_set.insert(5);
26     myunordered_set.insert(4);
27     myunordered_set.insert(5);
28     myunordered_set.insert(4);
29
30     for (const auto &s : myunordered_set) {
31         cout << s << " ";
32     }
33     cout << "\n";
34
35     return 0;
36 }

```

```

37
38 /*
39 output
40 4 5 2 1 3
41 */
42
43 unordered_set 刪除指定元素
44 #include <iostream>
45 #include <unordered_set>
46
47 int main() {
48     unordered_set<int> myunordered_set{2, 4, 6, 8};
49
50     myunordered_set.erase(2);
51     for (const auto &s : myunordered_set) {
52         cout << s << " ";
53     }
54     cout << "\n";
55 }

```

```

56     return 0;
57 }
58 /*
59 output
60 8 6 4
61 */
62
63 清空 unordered_set 元素
64 unordered_set<int> myunordered_set;
65 myunordered_set.insert(1);
66 myunordered_set.clear();
67
68 unordered_set 判斷元素是否存在
69 unordered_set<int> myunordered_set;
70 myunordered_set.insert(2);
71 myunordered_set.insert(4);
72 myunordered_set.insert(6);
73 cout << myunordered_set.count(4) << "\n"; // 1
74 cout << myunordered_set.count(8) << "\n"; // 0
75
76 判斷 unordered_set 容器是否為空
77 #include <iostream>
78 #include <unordered_set>
79
80 int main() {
81     unordered_set<int> myunordered_set;
82     myunordered_set.clear();
83
84     if (myunordered_set.empty()) {
85         cout << "empty\n";
86     } else {
87         cout << "not empty, size is " <<
            myunordered_set.size() << "\n";
88     }
89
90     return 0;
91 }

```

4.9 單調隊列

```

1 //單調隊列
2 "如果一個選手比你小還比你強，你就可以退役了。"--單調隊列
3
4 example 1
5
6 給出一個長度為 n 的數組，
7 輸出每 k 個連續的數中的最大值和最小值。
8
9 //寫法1
10 #include <bits/stdc++.h>
11 #define maxn 1000100
12 using namespace std;
13 int q[maxn], a[maxn];
14 int n, k;
15
16 void getmin() {
17     // 得到這個隊列裡的最小值，直接找到最後的就行了
18     int head = 0, tail = 0;
19     for (int i = 1; i < k; i++) {
20         while (head <= tail && a[q[tail]] >= a[i])
21             tail--;
22         q[++tail] = i;
23     }
24     for (int i = k; i <= n; i++) {
25         while (head <= tail && a[q[tail]] >= a[i])
26             tail--;
27         q[++tail] = i;
28         while (q[head] <= i - k) head++;
29         cout << a[q[head]] << " ";
30     }
31 }
32
33 void getmax() { // 和上面同理
34     int head = 0, tail = 0;
35     for (int i = 1; i < k; i++) {

```

```

34     while(head<=tail&&a[q[tail]]<=a[i])tail--;
35     q[++tail] = i;
36 }
37 for (int i = k; i <= n; i++) {
38     while(head<=tail&&a[q[tail]]<=a[i])tail--;
39     q[++tail] = i;
40     while (q[head] <= i - k) head++;
41     cout<<a[q[head]]<<" ";
42 }
43 }
44
45 int main() {
46     cin>>n>>k;    //每k個連續的數
47     for (int i = 1; i <= n; i++) cin>>a[i];
48     getmin();
49     cout<<"\n";
50     getmax();
51     cout<<"\n";
52     return 0;
53 }
54
55 //寫法2
56 #include <iostream>
57 #include <cstring>
58 #include <deque>
59 using namespace std;
60 int a[1000005];
61
62 int main() {
63     ios_base::sync_with_stdio(0);
64     int n, k;
65     while(cin>>n>>k) {
66         for(int i=0; i<n; i++) cin >> a[i];
67         deque<int> dq;
68         for(int i=0; i<n; i++){
69             while(dq.size() && dq.front()<=i-k)
70                 dq.pop_front();
71             while(dq.size() && a[dq.back()]>a[i])
72                 dq.pop_back();
73             dq.push_back(i);
74             if(i==k-1) cout<<a[dq.front()];
75             if(i>k-1) cout<<" "<<a[dq.front()];
76         }
77         if(k>n) cout<<a[dq.front()];
78         cout<<"\n";
79         while(dq.size()) dq.pop_back();
80         for(int i=0; i<n; i++){
81             while(dq.size() && dq.front()<=i-k)
82                 dq.pop_front();
83             while(dq.size() && a[dq.back()]<a[i])
84                 dq.pop_back();
85             dq.push_back(i);
86             if(i==k-1) cout<<a[dq.front()];
87             if(i>k-1) cout<<" "<<a[dq.front()];
88         }
89         if(k>n) cout<<a[dq.front()];
90         cout<<"\n";
91     }
92     return 0;
93 }

```

example 2

一個含有 n 項的數列，求出每一項前的 m 個數到它這個區間內的最小值。

若前面的數不足 m 項則從第 1 個數開始，若前面沒有數則輸出 0

```

100 #include<bits/stdc++.h>
101 using namespace std;
102 #define re register int
103 #define INF 0x3f3f3f3f
104 #define ll long long
105 #define maxn 2000009
106 #define maxm
107 #define ll read() {
108

```

```

109     ll x=0,f=1;
110     char ch=getchar();
111     while(ch<'0' || ch>'9'){
112         if(ch=='-') f=-1;
113         ch=getchar();
114     }
115     while(ch>='0' && ch<='9'){
116         x=(x<<1)+(x<<3)+(ll)(ch-'0');
117         ch=getchar();
118     }
119     return x*f;
120 }
121 int n,m,k,tot,head,tail;
122 int a[maxn],q[maxn];
123 int main() {
124     n=read(), m=read();
125     for(int i=1;i<=n;i++) a[i]=read();
126     head=1,tail=0; //起始位置為1
127     //因為插入是q[++tail]所以要初始化為0
128     for(int i=1;i<=n;i++)
129         //每次隊首的元素就是當前的答案
130     {
131         cout<<a[q[head]]<<endl;
132         while(i-q[head]+1>m&&head<=tail) //維護隊首
133             head++;
134         while(a[i]<a[q[tail]]&&head<=tail) //維護隊尾
135             tail--;
136         q[++tail]=i;
137     }
138     return 0;
139 }

```

5 sort

5.1 大數排序

```

1 #python大數排序
2
3 while True:
4     try:
5         n = int(input())          # 有幾筆數字需要排序
6         arr = []                  # 建立空串列
7         for i in range(n):
8             arr.append(int(input())) # 依序將數字存入串列
9         arr.sort()                 # 串列排序
10        for i in arr:
11            print(i)                # 依序印出串列中每個項目
12    except:
13        break

```

5.2 bubble sort

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main() {
5     int n;
6     cin>>n;
7     int a[n], tmp;
8     for(int i=0; i<n; i++) cin>>a[i];
9     for(int i=n-1; i>0; i--) {
10         for(int j=0; j<=i-1; j++) {
11             if( a[j]>a[j+1]) {
12                 tmp=a[j];
13                 a[j]=a[j+1];
14                 a[j+1]=tmp;
15             }
16         }
17     }
18     for(int i=0; i<n; i++) cout<<a[i]<<" ";
19 }

```

6 math

6.1 質數與因數

```

1 質數
2
3 埃氏篩法
4 int n;
5 vector<int> isprime(n+1,1);
6 isprime[0]=isprime[1]=0;
7 for(int i=2;i*i<=n;i++){
8     if(isprime[i])
9         for(int j=i*i;j<=n;j+=i) isprime[j]=0;
10 }
11
12 因數
13
14 最大公因數 O(log(min(a,b)))
15
16 int GCD(int a, int b)
17 {
18     if (b == 0) return a;
19     return GCD(b, a % b);
20 }
21
22 質因數分解
23
24 void primeFactorization(int n)
25 {
26     for (int i = 0; i < (int)p.size(); ++i)
27     {
28         if (p[i] * p[i] > n)
29             break;
30         if (n % p[i])
31             continue;
32         cout << p[i] << ' ';
33         while (n % p[i] == 0)
34             n /= p[i];
35     }
36     if (n != 1)
37         cout << n << ' ';
38     cout << '\n';
39 }
40
41 歌德巴赫猜想
42 solution : 把偶數 N (6≤N≤10^6) 寫成兩個質數的和。
43 #include <iostream>
44 #include <cstdio>
45 using namespace std;
46 #define N 2000000
47 int ox[N], p[N], pr;
48
49 void PrimeTable(){
50     ox[0] = ox[1] = 1;
51     pr = 0;
52     for (int i = 2; i < N; i++){
53         if (!ox[i]) p[pr++] = i;
54         for (int j = 0; i*p[j]<N&&j < pr; j++)
55             ox[i*p[j]] = 1;
56     }
57 }
58
59 int main(){
60     PrimeTable();
61     int n;
62     while (cin>>n,n){
63         int x;
64         for (x = 1;; x += 2)
65             if (!ox[x] && !ox[n - x])break;
66         printf("%d = %d + %d\n", n, x, n - x);
67     }
68 }
69
70 problem : 給定整數 N，求 N
    最少可以拆成多少個質數的和。

```

```

71 如果 N 是質數，則答案為 1。
72 如果 N 是偶數(不包含2)，則答案為 2 (強歌德巴赫猜想)。
73 如果 N 是奇數且 N-2 是質數，則答案為 2 (2+質數)。
74 其他狀況答案為 3 (弱歌德巴赫猜想)。
75 #pragma GCC optimize("O2")
76 #include <bits/stdc++.h>
77 using namespace std;
78 #define FOR(i, L, R) for(int i=L;i<(int)R;++i)
79 #define FORD(i, L, R) for(int i=L;i>(int)R;--i)
80 #define IOS
81     cin.tie(nullptr);
82     cout.tie(nullptr);
83     ios_base::sync_with_stdio(false);
84
85 bool isPrime(int n){
86     FOR(i, 2, n){
87         if (i * i > n)
88             return true;
89         if (n % i == 0)
90             return false;
91     }
92     return true;
93 }
94
95 int main(){
96     IOS;
97     int n;
98     cin >> n;
99     if(isPrime(n)) cout << "1\n";
100     else if(n%2==0||isPrime(n-2)) cout<<"2\n";
101     else cout << "3\n";
102 }

```

6.2 prime factorization

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main() {
5     int n;
6     while(true) {
7         cin>>n;
8         for(int x=2; x<=n; x++) {
9             while(n%x==0) {
10                 cout<<x<<"*";
11                 n/=x;
12             }
13         }
14         cout<<"\b \n";
15     }
16     system("pause");
17     return 0;
18 }

```

6.3 快速幂

```

1 計算a^b
2 #include <iostream>
3 #define ll long long
4 using namespace std;
5
6 const ll MOD = 1000000007;
7 ll fp(ll a, ll b) {
8     int ans = 1;
9     while(b > 0) {
10         if(b & 1) ans = ans * a % MOD;
11         a = a * a % MOD;
12         b >>= 1;
13     }
14     return ans;
15 }
16
17 int main() {

```

```

18 int a, b;
19 cin>>a>>b;
20 cout<<fp(a,b);
21 }

```

6.4 歐拉函數

```

1 //計算閉區間 [1,n] 中的正整數與 n 互質的個數
2 #include <bits/stdc++.h>
3 using namespace std;
4 int n,ans;
5
6 int phi(){
7     ans=n;
8     for(int i=2;i*i<=n;i++){
9         if(n%i==0){
10             ans=ans-ans/i;
11             while(n%i==0) n/=i;
12         }
13     }
14     if(n>1) ans=ans-ans/n;
15     return ans;
16 }
17 int main(){
18     while(cin>>n)
19         cout<<phi()<<endl;
20 }

```

7 algorithm

7.1 basic

```

1 min_element：找尋最小元素
2 min_element(first, last)
3 max_element：找尋最大元素
4 max_element(first, last)
5 sort：排序，預設由小排到大。
6 sort(first, last)
7 sort(first, last, cmp)：可自行定義比較運算子 cmp。
8 find：尋找元素。
9 find(first, last, val)
10 lower_bound：尋找第一個小於 x
    的元素位置，如果不存在，則回傳 last。
11 lower_bound(first, last, val)
12 upper_bound：尋找第一個大於 x
    的元素位置，如果不存在，則回傳 last。
13 upper_bound(first, last, val)
14 next_permutation：將序列順序轉換成下一個字典序，
    如果存在回傳 true，反之回傳 false。
15 next_permutation(first, last)
16 prev_permutation：將序列順序轉換成上一個字典序，
    如果存在回傳 true，反之回傳 false。
17 prev_permutation(first, last)

```

7.2 binarysearch

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int binary_search(vector<int> &nums, int target) {
5     int left=0, right=nums.size()-1;
6     while(left<=right){
7         int mid=(left+right)/2;
8         if (nums[mid]>target) right=mid-1;
9         else if (nums[mid]<target) left=mid+1;
10        else return mid+1;
11    }
12    return 0;
13 }

```

```

14
15 int main() {
16     int n, k, x;
17     cin >> n >> k;
18     int a[n];
19     vector<int> v;
20     for(int i=0 ; i<n ; i++){
21         cin >> x;
22         v.push_back(x);
23     }
24     for(int i=0 ; i<k ; i++) cin >> a[i];
25     for(int i=0 ; i<k ; i++){
26         cout << binary_search(v, a[i]) << endl;
27     }
28 }
29
30 lower_bound(a, a + n, k); //最左邊 ≥ k 的位置
31 upper_bound(a, a + n, k); //最左邊 > k 的位置
32 upper_bound(a, a + n, k) - 1; //最右邊 ≤ k 的位置
33 lower_bound(a, a + n, k) - 1; //最右邊 < k 的位置
34 (lower_bound, upper_bound) //等於 k 的範圍
35 equal_range(a, a+n, k);
36
37 /*
38 input
39 5 5
40 1 3 4 7 9
41 3 1 9 7 -2
42 */
43
44 /*
45 output
46 2
47 1
48 5
49 4
50 0
51 */

```

7.3 prefix sum

```

1 // 前綴和
2 陣列前n項的和。
3 b[i] = a[0] + a[1] + a[2] + ... + a[i]
4 區間和 [l, r] : b[r]-b[l-1] (要保留b[l]所以-1)
5
6 #include <bits/stdc++.h>
7 using namespace std;
8 int main(){
9     int n;
10    cin >> n;
11    int a[n], b[n];
12    for(int i=0; i<n; i++) cin >> a[i];
13    b[0] = a[0];
14    for(int i=1; i<n; i++) b[i] = b[i-1] + a[i];
15    for(int i=0; i<n; i++) cout<<b[i]<<' ';
16    cout<<'\\n';
17    int l, r;
18    cin >> l >> r;
19    cout << b[r] - b[l-1] ; //區間和
20 }

```

7.4 差分

```

1 // 差分
2 用途：在區間 [l, r] 加上一個數字v。
3 b[l] += v; (b[0~l] 加上v)
4 b[r+1] -= v; (b[r+1~n] 減去v (b[r] 仍保留v))
5 給的 a[] 是前綴和數列，建構 b[]，
6 因為 a[i] = b[0] + b[1] + b[2] + ... + b[i]，
7 所以 b[i] = a[i] - a[i-1]。
8 在 b[l] 加上 v，b[r+1] 減去 v，

```



```

9 最後再從 0 跑到 n 使 b[i] += b[i-1]。
10 這樣一來，b[] 是一個在某區間加上v的前綴和。
11
12 #include <bits/stdc++.h>
13 using namespace std;
14 int a[1000], b[1000];
15 // a: 前綴和數列, b: 差分數列
16 int main(){
17     int n, l, r, v;
18     cin >> n;
19     for(int i=1; i<=n; i++){
20         cin >> a[i];
21         b[i] = a[i] - a[i-1]; //建構差分數列
22     }
23     cin >> l >> r >> v;
24     b[l] += v;
25     b[r+1] -= v;
26
27     for(int i=1; i<=n; i++){
28         b[i] += b[i-1];
29         cout << b[i] << ' ';
30     }
31 }

```

7.5 greedy

```

1 //貪心
2 貪心演算法的核心為，
3 採取在目前狀態下最好或最佳（即最有利）的選擇。
4 貪心演算法雖然能獲得當前最佳解，
5 但不保證能獲得最後（全域）最佳解，
6 提出想法後可以先試圖尋找有沒有能推翻原本的想法的反例，
7 確認無誤再實作。

```

Scarecrow

//problem

有一個 $N \times 1$ 的稻田，有些稻田現在有種植作物，
為了避免被動物破壞，需要放置稻草人，
稻草人可以保護該塊稻田和左右兩塊稻田，
請問最少需要多少稻草人才能保護所有稻田？

//solution

從左到右掃描稻田，如果第 i 塊稻田有作物，
就把稻草人放到第 $i+1$ 塊稻田，
這樣能保護第 $i, i+1, i+2$ 塊稻田，
接著從第 $i+3$ 塊稻田繼續掃描。

//code

```

23 #include <bits/stdc++.h>
24 using namespace std;
25 int main(){
26     string s;
27     int i, n, t, tc = 1;
28     cin >> t;
29     while (t--){
30         cin >> n >> s;
31         int nc = 0;
32         for (i = 0; i < n; i++){
33             if (s[i] == '.') i += 2, nc++;
34             cout << "Case " << tc++ << ": " << nc << endl;
35         }
36     }
37 }

```

霍夫曼樹的變形題

//problem

給定 N 個數，每次將兩個數 a, b 合併成 $a+b$ ，
只到最後只剩一個數，合併成本為兩數和，
問最小合併成本為多少。

//solution

每次將最小的兩數合併起來。

//code

```

48 #include <bits/stdc++.h>
49 using namespace std;
50 int main()
51 {
52     int n, x;
53     while (cin >> n, n){
54         priority_queue<int, vector<int>, greater<int>>
55             q;
56         while (n--){
57             cin >> x;
58             q.push(x);
59         }
60         long long ans = 0;
61         while (q.size() > 1){
62             x = q.top();
63             q.pop();
64             x += q.top();
65             q.pop();
66             q.push(x);
67             ans += x;
68         }
69         cout << ans << endl;
70     }
71 }

```

Commando War

//problem

有 n 個部下，每個部下要花 B_i 分鐘交待任務，
再花 J_i 分鐘執行任務，一次只能對一位部下交代任務，
但可以多人同時執行任務，問最少要花多少時間完成任務。

//solution

執行時間長的人先交代任務

//code

```

81 #include <bits/stdc++.h>
82 using namespace std;
83 struct Data{
84     int b, j;
85     bool operator<(const Data &rhs) const {
86         return j > rhs.j;
87     }
88 };
89
90
91 int main(){
92     int n, ti = 0;
93     Data a[1005];
94     while (cin >> n, n){
95         for (int i = 0; i < n; ++i)
96             cin >> a[i].b >> a[i].j;
97         sort(a, a + n);
98         int ans = 0, sum = 0;
99         for (int i = 0; i < n; ++i){
100             sum += a[i].b;
101             ans = max(ans, sum + a[i].j);
102         }
103         cout << "Case " << ++ti << ": " << ans << '\n';
104     }
105 }

```

刪數字問題

//problem

給定一個數字 $N (\leq 10^{100})$ ，需要刪除 K 個數字，
請問刪除 K 個數字後最小的數字為何？

//solution

刪除滿足第 i 位數大於第 $i+1$ 位數的最左邊第 i 位數，
扣除高位數的影響較扣除低位數的大。

//code

```

116 int main()
117 {
118     string s;
119     int k;
120     cin >> s >> k;
121     for (int i = 0; i < k; ++i){

```



```

123     if ((int)s.size() == 0) break;
124     int pos = (int)s.size() - 1;
125     for (int j = 0; j < (int)s.size() - 1; ++j){
126         if (s[j] > s[j + 1]){
127             pos = j;
128             break;
129         }
130     }
131     s.erase(pos, 1);
132 }
133 while ((int)s.size() > 0 && s[0] == '0')
134     s.erase(0, 1);
135 if ((int)s.size()) cout << s << '\n';
136 else cout << 0 << '\n';
137 }
138
139

```

區間覆蓋長度

//problem

給定 n 條線段區間為 $[Li, Ri]$ ，

請問這些線段的覆蓋所覆蓋的長度？

144

//solution

先將所有區間依照左界由小到大排序，

左界相同依照右界由小到大排序，

用一個變數 R 紀錄目前最大可以覆蓋到的右界。

如果目前區間左界 $\leq R$ ，代表該區間可以和前面的線段合併。

150

//code

struct Line

```

153 {
154     int L, R;
155     bool operator<(const Line &rhs) const
156     {
157         if (L != rhs.L) return L < rhs.L;
158         return R < rhs.R;
159     }
160 };
161

```

int main(){

```

163     int n;
164     Line a[10005];
165     while (cin >> n){
166         for (int i = 0; i < n; i++)
167             cin >> a[i].L >> a[i].R;
168         sort(a, a + n);
169         int ans = 0, L = a[0].L, R = a[0].R;
170         for (int i = 1; i < n; i++){
171             if (a[i].L < R) R = max(R, a[i].R);
172             else{
173                 ans += R - L;
174                 L = a[i].L;
175                 R = a[i].R;
176             }
177         }
178         cout << ans + (R - L) << '\n';
179     }
180 }
181
182

```

最小區間覆蓋長度

//problem

給定 n 條線段區間為 $[Li, Ri]$ ，

請問最少要選幾個區間才能完全覆蓋 $[0, S]$ ？

187

//solution

先將所有區間依照左界由小到大排序，

對於當前區間 $[Li, Ri]$ ，要從左界 $> Ri$ 的所有區間中，

找到有著最大的右界的區間，連接當前區間。

192

//problem

長度 n 的直線中有數個加熱器，

在 x 的加熱器可以讓 $[x-r, x+r]$ 內的物品加熱，

問最少要幾個加熱器可以把 $[0, n]$ 的範圍加熱。

197

//solution

199 對於最左邊沒加熱的點 a ，選擇最遠可以加熱 a 的加熱器，
200 更新已加熱範圍，重複上述動作繼續尋找加熱器。

201

//code

```

202 int main(){
203     int n, r;
204     int a[1005];
205     cin >> n >> r;
206     for (int i=1; i<=n; ++i) cin>>a[i];
207     int i = 1, ans = 0;
208     while (i <= n){
209         int R=min(i+r-1, n), L=max(i-r+1, 0)
210         int nextR=-1;
211         for (int j = R; j >= L; --j){
212             if (a[j]){
213                 nextR = j;
214                 break;
215             }
216         }
217         if (nextR == -1){
218             ans = -1;
219             break;
220         }
221         ++ans;
222         i = nextR + r;
223     }
224     cout << ans << '\n';
225 }
226
227
228

```

最多不重疊區間

//problem

給你 n 條線段區間為 $[Li, Ri]$ ，

請問最多可以選擇幾條不重疊的線段(頭尾可相連)？

233

//solution

依照右界由小到大排序，

每次取到一個不重疊的線段，答案 $+1$ 。

236

//code

struct Line

```

240 {
241     int L, R;
242     bool operator<(const Line &rhs) const {
243         return R < rhs.R;
244     }
245 };
246
247 int main(){
248     int t;
249     cin >> t;
250     Line a[30];
251     while (t--){
252         int n = 0;
253         while (cin>>a[n].L>>a[n].R, a[n].L||a[n].R)
254             ++n;
255         sort(a, a + n);
256         int ans = 1, R = a[0].R;
257         for (int i = 1; i < n; i++){
258             if (a[i].L >= R){
259                 ++ans;
260                 R = a[i].R;
261             }
262         }
263         cout << ans << '\n';
264     }
265 }
266
267

```

區間選點問題

//problem

給你 n 條線段區間為 $[Li, Ri]$ ，

請問至少要取幾個點才能讓每個區間至少包含一個點？

272

//solution

將區間依照右界由小到大排序， R =第一個區間的右界，

274

```

275 遍歷所有區段，如果當前區間左界>R，
276 代表必須多選一個點 (ans+=1)，並將 R=當前區間右界。
277
278 //problem
279 給定 N 個座標，要在 x 軸找到最小的點，
280 讓每個座標至少和一個點距離 ≤ D。
281
282 //solution
283 以每個點 (xi,yi) 為圓心半徑為 D 的圓 c，
284 求出 C 和 x 軸的交點 Li,Ri，題目轉變成區間選點問題。
285
286 //code
287 struct Line
288 {
289     int L, R;
290     bool operator<(const Line &rhs) const {
291         return R < rhs.R;
292     }
293 };
294
295 int main(){
296     int t;
297     cin >> t;
298     Line a[30];
299     while (t--){
300         int n = 0;
301         while (cin>>a[n].L>>a[n].R, a[n].L||a[n].R)
302             ++n;
303         sort(a, a + n);
304         int ans = 1, R = a[0].R;
305         for (int i = 1; i < n; i++){
306             if (a[i].L >= R){
307                 ++ans;
308                 R = a[i].R;
309             }
310         }
311         cout << ans << '\n';
312     }
313 }
314
315 最小化最大延遲問題
316 //problem
317 給定 N 項工作，每項工作的需要處理時長為 Ti，
318 期限是 Di，第 i 項工作延遲的時間為 Li=max(0,Fi-Di)，
319 原本Fi 為第 i 項工作的完成時間，
320 求一種工作排序使 maxLi 最小。
321
322 //solution
323 按照到期時間從早到晚處理。
324
325 //code
326 struct Work
327 {
328     int t, d;
329     bool operator<(const Work &rhs) const {
330         return d < rhs.d;
331     }
332 };
333 };
334
335 int main(){
336     int n;
337     Work a[10000];
338     cin >> n;
339     for (int i = 0; i < n; ++i)
340         cin >> a[i].t >> a[i].d;
341     sort(a, a + n);
342     int maxL = 0, sumT = 0;
343     for (int i = 0; i < n; ++i){
344         sumT += a[i].t;
345         maxL = max(maxL, sumT - a[i].d);
346     }
347     cout << maxL << '\n';
348 }
349
350

```

```

351 最少延遲數量問題
352 //problem
353 給定 N 個工作，每個工作的需要處理時長為 Ti，
354 期限是 Di，求一種工作排序使得逾期工作數量最小。
355
356 //solution
357 期限越早到期的工作越先做。將工作依照到期時間從早到晚排序，
358 依序放入工作列表中，如果發現有工作預期，
359 就從目前選擇的工作中，移除耗時最長的工作。
360
361 上述方法為 Moore-Hodgson's Algorithm。
362
363 //problem
364 給定烏龜的重量和可承受重量，問最多可以疊幾隻烏龜？
365
366 和最少延遲數量問題是相同的問題，只要將題敘做轉換。
367
368 工作處理時長 → 烏龜重量
369 工作期限 → 烏龜可承受重量
370 多少工作不延期 → 可以疊幾隻烏龜
371
372 //code
373 struct Work{
374     int t, d;
375     bool operator<(const Work &rhs) const {
376         return d < rhs.d;
377     }
378 };
379
380 int main(){
381     int n = 0;
382     Work a[10000];
383     priority_queue<int> pq;
384     while(cin >> a[n].t >> a[n].d)
385         ++n;
386     sort(a, a + n);
387     int sumT = 0, ans = n;
388     for (int i = 0; i < n; ++i){
389         pq.push(a[i].t);
390         sumT += a[i].t;
391         if(a[i].d<sumT){
392             int x = pq.top();
393             pq.pop();
394             sumT -= x;
395             --ans;
396         }
397     }
398     cout << ans << '\n';
399 }
400
401 任務調度問題
402 //problem
403 給定 N 項工作，每項工作的需要處理時長為 Ti，
404 期限是 Di，如果第 i 項工作延遲需要受到 pi 單位懲罰，
405 請問最少會受到多少單位懲罰。
406
407 //solution
408 依照懲罰由大到小排序，
409 每項工作依序嘗試可不可以放在 Di-Ti+1,Di-Ti,...,1,0，
410 如果有空閒就放進去，否則延後執行。
411
412 //problem
413 給定 N 項工作，每項工作的需要處理時長為 Ti，
414 期限是 Di，如果第 i 項工作在期限內完成會獲得 ai
    單位獎勵，
415 請問最多會獲得多少單位獎勵。
416
417 //solution
418 和上題相似，這題變成依照獎勵由大到小排序。
419
420 //code
421 struct Work
422 {
423     int d, p;
424     bool operator<(const Work &rhs) const {

```

```

425     return p > rhs.p;
426 }
427 };
428
429 int main(){
430     int n;
431     Work a[100005];
432     bitset<100005> ok;
433     while (cin >> n){
434         ok.reset();
435         for (int i = 0; i < n; ++i)
436             cin >> a[i].d >> a[i].p;
437         sort(a, a + n);
438         int ans = 0;
439         for (int i = 0; i < n; ++i){
440             int j = a[i].d;
441             while (j--){
442                 if (!ok[j]){
443                     ans += a[i].p;
444                     ok[j] = true;
445                     break;
446                 }
447             }
448             cout << ans << '\n';
449         }
450     }
451 }
452
453 //problem
454 給定 N 項工作，每項工作的需要處理時長為  $T_i$ ，
455 有 M 台機器可執行多項工作，但不能將工作拆分，
456 最快可以在什麼時候完成所有工作？
457
458 //solution
459 將工作由大到小排序，每項工作交給最快空閒的機器。
460
461 //code
462 int main(){
463     int n, m;
464     int a[10000];
465     cin >> n >> m;
466     for (int i = 0; i < n; ++i)
467         cin >> a[i];
468     sort(a, a + n, greater<int>());
469     int ans = 0;
470     priority_queue<int, vector<int>, greater<int>> pq;
471     for (int i = 0; i < m && i < n; ++i){
472         ans = max(ans, a[i]);
473         pq.push(a[i]);
474     }
475     for (int i = m; i < n; ++i){
476         int x = pq.top();
477         pq.pop();
478         x += a[i];
479         ans = max(ans, x);
480         pq.push(x);
481     }
482     cout << ans << '\n';
483 }

```

8 動態規劃

8.1 LCS 和 LIS

```

1 //最長共同子序列(LCS)
2 給定兩序列 A,B，求最長的序列 C，
3 C 同時為 A,B 的子序列。
4
5 //最長遞增子序列(LIS)
6 給你一個序列 A，求最長的序列 B，
7 B 是一個（非）嚴格遞增序列，且為 A 的子序列。
8
9 //LCS 和 LIS 題目轉換

```

```

10 LIS 轉成 LCS
11 1. A 為原序列，B=sort(A)
12 2. 對 A,B 做 LCS
13 LCS 轉成 LIS
14 1. A, B 為原本的兩序列
15 2. 最 A 序列作編號轉換，將轉換規則套用在 B
16 3. 對 B 做 LIS
17 4. 重複的數字在編號轉換時後要變成不同的數字，
18 越早出現的數字要越小
19 5. 如果有數字在 B 裡面而不在 A 裡面，
20 直接忽略這個數字不做轉換即可

```

9 graph

9.1 graph

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 class Node {
5 public:
6     int val;
7     vector<Node*> children;
8
9     Node() {}
10
11     Node(int _val) {
12         val = _val;
13     }
14
15     Node(int _val, vector<Node*> _children) {
16         val = _val;
17         children = _children;
18     }
19 };
20
21 struct ListNode {
22     int val;
23     ListNode *next;
24     ListNode() : val(0), next(nullptr) {}
25     ListNode(int x) : val(x), next(nullptr) {}
26     ListNode(int x, ListNode *next) : val(x),
27         next(next) {}
28 };
29
30 struct TreeNode {
31     int val;
32     TreeNode *left;
33     TreeNode *right;
34     TreeNode() : val(0), left(nullptr),
35         right(nullptr) {}
36     TreeNode(int x) : val(x), left(nullptr),
37         right(nullptr) {}
38     TreeNode(int x, TreeNode *left, TreeNode *right)
39         : val(x), left(left), right(right) {}
40 };
41
42 class ListProblem {
43     vector<int> nums={};
44 public:
45     void solve() {
46         return;
47     }
48
49     ListNode* buildList(int idx) {
50         if(idx == nums.size()) return NULL;
51         ListNode *current=new
52             ListNode(nums[idx++],current->next);
53         return current;
54     }
55
56     void deleteList(ListNode* root) {
57         if(root == NULL) return;
58     }
59 }

```

```

53     deleteList(root->next);
54     delete root;
55     return;
56 }
57 };
58
59 class TreeProblem {
60     int null = INT_MIN;
61     vector<int> nums = {}, result;
62 public:
63     void solve() {
64
65         return;
66     }
67
68     TreeNode* buildBinaryTreeUsingDFS(int left, int
        right) {
69         if((left > right) || (nums[(left+right)/2] ==
            null)) return NULL;
70         int mid = (left+right)/2;
71         TreeNode* current = new TreeNode(
            nums[mid],
72             buildBinaryTreeUsingDFS(left,mid-1),
73             buildBinaryTreeUsingDFS(mid+1,right));
74         return current;
75     }
76 }
77
78     TreeNode* buildBinaryTreeUsingBFS() {
79         int idx = 0;
80         TreeNode* root = new TreeNode(nums[idx++]);
81         queue<TreeNode*> q;
82         q.push(root);
83         while(idx < nums.size()) {
84             if(nums[idx] != null) {
85                 TreeNode* left = new
                    TreeNode(nums[idx]);
86                 q.front()->left = left;
87                 q.push(left);
88             }
89             idx++;
90             if((idx < nums.size()) && (nums[idx] !=
                null)) {
91                 TreeNode* right = new
                    TreeNode(nums[idx]);
92                 q.front()->right = right;
93                 q.push(right);
94             }
95             idx++;
96             q.pop();
97         }
98         return root;
99     }
100
101     Node* buildNaryTree() {
102         int idx = 2;
103         Node *root = new Node(nums.front());
104         queue<Node*> q;
105         q.push(root);
106         while(idx < nums.size()) {
107             while((idx < nums.size()) && (nums[idx]
                != null)) {
108                 Node *current = new Node(nums[idx++]);
109                 q.front()->children.push_back(current);
110                 q.push(current);
111             }
112             idx++;
113             q.pop();
114         }
115         return root;
116     }
117
118     void deleteBinaryTree(TreeNode* root) {
119         if(root->left != NULL)
            deleteBinaryTree(root->left);
120         if(root->right != NULL)
            deleteBinaryTree(root->right);
121         delete root;

```

```

122         return;
123     }
124
125     void deleteNaryTree(Node* root) {
126         if(root == NULL) return;
127         for(int i=0; i<root->children.size(); i++) {
128             deleteNaryTree(root->children[i]);
129             delete root->children[i];
130         }
131         delete root;
132         return;
133     }
134
135     void inorderTraversal(TreeNode* root) {
136         if(root == NULL) return;
137         inorderTraversal(root->left);
138         cout<<root->val<< ' ';
139         inorderTraversal(root->right);
140         return;
141     }
142 };
143
144 int main() {
145
146     return 0;
147 }

```

10 Section2

10.1 thm

• 中文測試

$$\cdot \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$

11 space

11.1 s

```

1 /*
2 1
3 2
4 3
5 4
6 5
7 6
8 7
9 8
10 9
11 10
12 11
13 12
14 13
15 14
16 15
17 16
18 17
19 18
20 19
21 20
22 21
23 22
24 23
25 24
26 25
27 26
28 27
29 28
30 29
31 30
32 31
33 */

```

12 reference

12.1 assert.cpp

```

1
2 template <typename T>
3 bool AssertVectorIsSorted(vector<T>& vec){
4     for(int i=0;i<(int)vec.size()-1;++i)
5         if(vec[i]>vec[i+1]) return false;
6     return true;
7 }

```

12.2 print.cpp

```

1 template <typename T>
2 void printvector(vector<T>& vec){
3     for(T &x:vec) cout<<x<<" ";
4     cout<<"\n";
5     return;
6 }

```

12.3 rvec.cpp

```

1 template <typename T = int>
2 vector<T> randomvec(int n){
3     vector<T> vec(n);
4     unsigned seed=chrono::system_clock::now()
5         .time_since_epoch().count();
6     default_random_engine generator(seed);
7     uniform_real_distribution<double>
8         distribution(0.0,200.0);
9     for(T &num:vec) num=distribution(generator);
10    return vec;

```

12.4 find.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 int arr[10000];
5
6 int main(){
7     string s,a,b;
8     int cnt=0;
9     while(getline(cin,s)){
10         if(cnt++) cout<<endl;
11         cin>>a>>ws>>b;
12         int pos=0,i=0,r=0;
13         memset(arr,0,sizeof(arr));
14         while(1){
15             pos=s.find(a,pos);
16             if(pos==-1) break;
17             arr[i++]=pos-r;
18             r+=b.size()-a.size();
19             s.replace(pos,a.length(),b);
20             pos+=b.length();
21         }
22         if(i)
23             for(int j=0;j<i;j++){
24                 if(j) cout<<" ";
25                 cout<<arr[j];
26             }
27         else cout<<-1;
28         cout<<endl<<s<<endl;
29         cin>>ws;
30     }
31     return 0;
32 }

```

12.5 sort.cpp

```

1 #include<iostream>
2 #include<vector>
3 using namespace std;
4
5 template<typename T>
6 int partition(vector<T>& arr,int low,int high) {
7     T pivot=arr[high];
8     int i=low-1;
9     for(int j=low;j<high;j++) {
10         if(arr[j]<pivot){
11             i++;
12             swap(arr[i],arr[j]);
13         }
14     }
15     swap(arr[i+1],arr[high]);
16     return i+1;
17 }
18
19 template<typename T>
20 void quickSort(vector<T>& arr,int low,int high) {
21     if(low<high){
22         int pivotIndex=partition(arr,low,high);
23         quickSort(arr,low,pivotIndex-1);
24         quickSort(arr,pivotIndex+1,high);
25     }
26 }
27
28 template<typename T>
29 void customSort(vector<T>& arr) {
30     int n=arr.size();
31     quickSort(arr,0,n-1);
32 }
33
34 int main(){
35     vector<int> numbers={5, 2, 8, 1, 3};
36     cout<<"Before sorting: ";
37     for(const auto& num:numbers)
38         cout<<num<<" ";
39     customSort(numbers);
40     cout<<"\nAfter sorting: ";
41     for(const auto& num : numbers)
42         cout<<num<<" ";
43     return 0;
44 }

```

12.6 minmax.cpp

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5 int main() {
6     vector<int> num
7         = {4, 2, 8, 5, 1, 9, 6, 3, 7};
8     sort(num.begin(), num.end());
9     int target = 5;
10    vector<int>::iterator it=
11        lower_bound(num.begin(),num.end(),target);
12    if (it!=num.end()&&*it==target) {
13        int index=distance(num.begin(), it);
14        cout<<target<<" 在  "<<index<<endl;
15    }
16    else
17        cout<<target<<" 未找到！ "<<endl;
18    pair<vector<int>::iterator, vector<int>::iterator>
19        minmaxElement = minmax_element(num.begin(),
20            num.end());
21    cout<<"最小元素： "<<*(minmaxElement.first)<<endl;
22    cout<<"最大元素： "<<*(minmaxElement.second)<<endl;
23    return 0;
24 }

```

12.7 hw6.cpp

```

1 #include<iostream>
2 #include<string>
3 #include<cstring>
4 #include<algorithm>
5 using namespace std;
6
7 class HugeInt{
8 private:
9     short integer[40];
10
11 public:
12     HugeInt(const string& s){
13         memset(integer,0,sizeof(integer));
14         for(int i=0;i<s.length();i++){
15             integer[i]=s[s.length()-1-i]-'0';
16         }
17
18     HugeInt operator+(const HugeInt& other)const{
19         HugeInt result("");
20         for(int i=0;i<40;i++){
21             result.integer[i]+=
22                 integer[i]+other.integer[i];
23             if(result.integer[i]>=10){
24                 result.integer[i]-=10;
25                 result.integer[i+1]++;
26             }
27         }
28         return result;
29     }
30
31     HugeInt operator-(const HugeInt& other)const{
32         HugeInt result("");
33         for(int i=0;i<40;i++){
34             result.integer[i]+=
35                 integer[i]-other.integer[i];
36             if(result.integer[i]<0){
37                 result.integer[i]+=10;
38                 result.integer[i+1]--;
39             }
40         }
41         return result;
42     }
43
44     HugeInt operator*(const HugeInt& other)const{
45         HugeInt result("");
46         for(int i=0;i<40;i++){
47             for(int j=0;j<40;j++){
48                 result.integer[i+j]+=
49                     integer[i]*other.integer[j];
50             }
51             for(int i=0;i<40;i++){
52                 result.integer[i+1]+=result.integer[i]/10;
53                 result.integer[i]%=10;
54             }
55             return result;
56         }
57
58     HugeInt operator/(const HugeInt& other)const{
59         HugeInt result("");
60         HugeInt remainder("0");
61         HugeInt num("10");
62         for(int i=39;i>=0;i--){
63             if(i==39) remainder.integer[0]=integer[i];
64             else{
65                 remainder=remainder*num;
66                 remainder.integer[0]=integer[i];
67             }
68             int quotient=0;
69             while(remainder>other){
70                 remainder=remainder-other;
71                 quotient++;
72             }
73             if(remainder==other){
74                 remainder=remainder-other;
75                 quotient++;
76
77                 result.integer[i]=quotient;
78             }
79             return result;
80         }
81
82     HugeInt operator%(const HugeInt& other)const{
83         HugeInt remainder("0");
84         HugeInt value("0");
85         HugeInt num("10");
86         for(int i=39;i>=0;i--){
87             remainder=remainder*num;
88             remainder.integer[0]=integer[i];
89             while(remainder>other)
90                 remainder=remainder-other;
91             if(remainder==other) remainder=value;
92         }
93         return remainder;
94     }
95
96     bool operator>(const HugeInt& other)const{
97         for(int i=39;i>=0;i--){
98             if(integer[i]>other.integer[i])
99                 return true;
100             else if(integer[i]<other.integer[i])
101                 return false;
102         }
103         return false;
104     }
105
106     bool operator<(const HugeInt& other)const{
107         for(int i=39;i>=0;i--){
108             if(integer[i]<other.integer[i])
109                 return true;
110             else if(integer[i]>other.integer[i])
111                 return false;
112         }
113         return false;
114     }
115
116     bool operator==(const HugeInt& other)const{
117         for(int i=39;i>=0;i--){
118             if(integer[i]!=other.integer[i])
119                 return false;
120         }
121         return true;
122     }
123
124     bool operator>=(const HugeInt& other)const{
125         for(int i=39;i>=0;i--){
126             if(integer[i]<other.integer[i])
127                 return false;
128             return true;
129         }
130
131     bool operator<=(const HugeInt& other)const{
132         for(int i=39;i>=0;i--){
133             if(integer[i]>other.integer[i])
134                 return false;
135             return true;
136         }
137
138     bool operator!=(const HugeInt& other)const{
139         return !(*this==other);
140     }
141
142     friend istream& operator>>
143     (istream& in,HugeInt& hugeInt){
144         string s;
145         in>>s;
146         hugeInt=HugeInt(s);
147         return in;
148     }
149
150     friend ostream& operator<<
151     (ostream& out,const HugeInt& hugeInt){
152         bool isLeadingZero=true;
153         for(int i=39;i>=0;i--){
154             if(hugeInt.integer[i])

```

```

153         isLeadingZero=false;
154         if(!isLeadingZero)
155             out<<hugeInt.integer[i];
156     }
157     if(isLeadingZero) out<<0;
158     return out;
159 }
160
161 void print(HugeInt a,HugeInt b){
162     if(a>b) cout<<a<<" > "<<b<<endl;
163     else if(a<b) cout<<a<<" < "<<b<<endl;
164     else cout<<a<<" = "<<b<<endl;
165     cout<<a<<" + "<<b<<" = "<<a+b<<endl;
166     if(a>b) cout<<a<<" - "<<b<<" = "<<a-b<<endl;
167     else if(a<b)
168         cout<<a<<" - "<<b<<" = -"<<b-a<<endl;
169     else cout<<a<<" - "<<b<<" = "<<a-b<<endl;
170     cout<<a<<" * "<<b<<" = "<<a*b<<endl;
171     cout<<a<<" / "<<b<<" = "<<a/b<<endl;
172     cout<<a<<" % "<<b<<" = "<<a%b<<endl;
173 }
174 };
175
176 int main(){
177     string x,y;
178     int cnt=0;
179     while(cin>>x>>y){
180         HugeInt x1(x);
181         if(cnt++) cout<<endl;
182         x1.print(x,y);
183     }
184 }

```

12.8 rectangle.cpp

```

1 #include<iostream>
2 #include<iomanip>
3 #include<algorithm>
4 #include<math.h>
5 using namespace std;
6 class R {
7     private:
8         double area,diagonal;
9     public:
10         R(){};
11         int length,width;
12         R(double l,double w) {
13             length=l;
14             width=w;
15         }
16         int getA(){
17             area=length*width;
18             return area;
19         }
20         double getD(){
21             diagonal=sqrt(length*length+width*width);
22             return diagonal;
23         }
24 };
25 bool cmp(R r1, R r2) {
26     if(r1.getA() == r2.getA())
27         return r1.getD()>r2.getD();
28     return r1.getA()<r2.getA();
29 }
30 int main() {
31     int n;
32     cin>>n;
33     R* r=new R[n];
34     for(int i=0;i<n;i++) {
35         int l,w;
36         cin>>l>>w;
37         r[i]=R(l,w);
38     }
39     sort(r,r+n,cmp);
40     for(int i=0;i<n;i++){
41         cout<<i+1<<" : ("<<r[i].length<<" , "<<r[i].width;

```

```

42         cout<<") area= "<<r[i].getA();
43         cout<<fixed<<setprecision(3)
44             <<" diagonal= "<<r[i].getD()<<endl;
45     }
46     delete[] r;
47     return 0;
48 }

```

12.9 SortFunctionTemplate.cpp

```

1 #include<iostream>
2 #include<vector>
3 using namespace std;
4
5 template<typename T>
6 int p(vector<T>& arr,int low,int high){
7     T pivot=arr[high];
8     int i=low-1;
9     for(int j=low;j<high;j++){
10         if(arr[j]<pivot){
11             i++;
12             swap(arr[i],arr[j]);
13         }
14         swap(arr[i+1],arr[high]);
15         return 1+i;
16     }
17
18 template<typename T>
19 void qsort(vector<T>& arr, int low,int high){
20     if(low<high){
21         int pindex=p(arr,low,high);
22         qsort(arr,low,pindex-1);
23         qsort(arr,pindex+1,high);
24     }
25 }
26
27 template<typename T>
28 void mysort(vector<T>& arr){
29     int n=arr.size();
30     qsort(arr,0,n-1);
31 }
32
33 template<typename T>
34 void solve(vector<T>& arr){
35     mysort(arr);
36     int cnt=0;
37     for(auto a:arr){
38         if(cnt++) cout<<" ";
39         cout<<a;
40     }
41     cout<<endl;
42     arr.clear();
43 }
44
45 int main(){
46     int n,x;
47     double d;
48     char c;
49     string s;
50     while(cin>>n){
51         vector<int> v;
52         for(int i=0;i<n;i++){
53             cin>>x;
54             v.push_back(x);
55         }
56         solve(v);
57         cin>>n;
58         vector<double> vd;
59         for(int i=0;i<n;i++){
60             cin>>d;
61             vd.push_back(d);
62         }
63         solve(vd);
64         cin>>n;
65         vector<char> vc;
66         for(int i=0;i<n;i++){

```



```

67         cin>>c;
68         vc.push_back(c);
69     }
70     solve(vc);
71     cin>>n>>ws;
72     vector<string> vs;
73     for(int i=0;i<n;i++){
74         getline(cin,s);
75         vs.push_back(s);
76     }
77     mysort(vs);
78     for(auto a:vs) cout<<a<<endl;
79     vs.clear();
80 }
81 return 0;
82 }

```

12.10 exception.cpp

```

1 //分母為0
2 #include <iostream>
3 #include <exception>
4 using namespace std;
5 double divide(int numerator, int denominator) {
6     if (denominator == 0) {
7         throw runtime_error("Division by zero!");
8     }
9     return static_cast<double> numerator/denominator;
10 }
11
12 int main() {
13     int a, b;
14     cout << "Enter two numbers: ";
15     cin >> a >> b;
16     try {
17         double result = divide(a, b);
18         cout << "Result: " << result << endl;
19     }
20     catch (const exception& ex) {
21         cout<<"Exception: "<<ex.what()<<endl;
22     }
23     return 0;
24 }
25
26 //自訂狀況類別以文字描述錯誤訊息
27 #include <iostream>
28 #include <exception>
29 using namespace std;
30 class MyException : public exception{
31 private:
32     string errorMessage;
33 public:
34     MyException(const string& message):
35         errorMessage(message) {}
36     const char* what() const throw(){
37         return errorMessage.c_str();
38     }
39 };
40 int main(){
41     try{
42         throw MyException("Custom exception
43             occurred!");
44     }
45     catch (const MyException& ex){
46         cout << "Exception: " << ex.what() << endl;
47     }
48     return 0;
49 }
50
51 //自訂狀況類別以代碼和文字描述錯誤訊息
52 #include <iostream>
53 #include <exception>
54 using namespace std;
55 class MyException : public exception{
56 private:

```

```

56     int errorCode;
57     string errorMessage;
58 public:
59     MyException(int code, const string& message)
60         : errorCode(code), errorMessage(message) {}
61     int getCode() const{
62         return errorCode;
63     }
64     const char* what() const throw(){
65         return errorMessage.c_str();
66     }
67 };
68 int main(){
69     try{
70         throw MyException(404, "Page not found!");
71     }
72     catch (const MyException& ex){
73         cout << "Exception: Error Code "
74             <<ex.getCode()<<" - "<<ex.what()<<endl;
75     }
76     return 0;
77 }
78
79 //自訂狀況類別使用更詳細的錯誤資訊
80 #include <iostream>
81 #include <exception>
82 using namespace std;
83 class MyException : public exception{
84 private:
85     string fileName;
86     int lineNumber;
87     string errorMessage;
88 public:
89     MyException(const string& file,
90         int line, const string& message)
91         : fileName(file), lineNumber(line),
92         errorMessage(message) {}
93
94     const string& getFile() const{
95         return fileName;
96     }
97
98     int getLine() const{
99         return lineNumber;
100     }
101
102     const char* what() const throw(){
103         return errorMessage.c_str();
104     }
105 };
106 int main(){
107     try{
108         throw MyException(__FILE__, __LINE__,
109             "Custom exception occurred!");
110     }
111     catch (const MyException& ex){
112         cout << "Exception: " << ex.what() << endl;
113         cout << "File: " << ex.getFile() << endl;
114         cout << "Line: " << ex.getLine() << endl;
115     }
116     return 0;
117 }

```

12.11 RationalNumber.cpp

```

1 #include<iostream>
2 #include<math.h>
3 #include<algorithm>
4 #include<vector>
5 #include<complex>
6 #include<iomanip>
7 using namespace std;
8
9 int gcd(int a,int b){
10     if(b==0) return a;

```

```

11     return gcd(b,a%b);
12 }
13
14 class Rational{
15 public:
16     int mol,den;
17     Rational(int m=1,int d=2){
18         mol=m;
19         den=d;
20     }
21
22     friend ostream& operator<<(ostream& os, const
23         Rational& r){
24         if(r.mol==0) os<<0;
25         else if(r.mol==1&&r.den==1) os<<1;
26         else os<<"("<<r.mol<<"/ " <<r.den<<")";
27         return os;
28     }
29
30     friend istream& operator>>(istream& is, Rational&
31         r){
32         char slash;
33         is>>r.mol>>slash>>r.den;
34         return is;
35     }
36
37     Rational operator+(const Rational& other){
38         Rational result;
39         result.mol=mol*other.den+other.mol*den;
40         result.den=den*other.den;
41         int com=gcd(result.mol,result.den);
42         result.mol/=com;
43         result.den/=com;
44         return result;
45     }
46
47     Rational operator-(const Rational& other) {
48         Rational result;
49         result.mol=mol*other.den-other.mol*den;
50         result.den=den*other.den;
51         int com=gcd(result.mol,result.den);
52         result.mol/=com;
53         result.den/=com;
54         return result;
55     }
56
57     Rational operator*(const Rational& other) {
58         Rational result;
59         result.mol=mol*other.mol;
60         result.den=den*other.den;
61         int com=gcd(result.mol,result.den);
62         result.mol/=com;
63         result.den/=com;
64         return result;
65     }
66
67     Rational operator/(const Rational& other) {
68         Rational result;
69         result.mol=mol*other.den;
70         result.den=den*other.mol;
71         int com=gcd(result.mol,result.den);
72         result.mol/=com;
73         result.den/=com;
74         return result;
75     }
76 };
77
78 int main(){
79     char op,g;
80     Rational r1,r2;
81     while(cin>>op){
82         cin>>g>>r1>>g>>r2>>g;
83         if(r1.mol%r1.den) cout<<r1;
84         else cout<<r1.mol/r1.den;
85         cout<<" "<<op<<" ";
86         if(r2.mol%r2.den) cout<<r2;

```

```

86         else cout<<r2.mol/r2.den;
87         cout<<" = ";
88         switch(op){
89             case '+':
90                 cout<<r1+r2<<endl;
91                 break;
92             case '-':
93                 cout<<r1-r2<<endl;
94                 break;
95             case '*':
96                 cout<<r1*r2<<endl;
97                 break;
98             case '/':
99                 cout<<r1/r2<<endl;
100                 break;
101         }
102     }
103     return 0;
104 }

```

12.12 purevirtual.cpp

```

1 #include <iostream>
2 using namespace std;
3
4 class Shape{
5 public:
6     virtual double getArea() const=0;
7 };
8
9 class Circle:public Shape{
10 private:
11     double radius;
12 public:
13     Circle(double r):radius(r) {}
14     double getArea() const override {
15         return 3.14159*radius*radius;
16     }
17 };
18
19 class Rectangle:public Shape{
20 private:
21     double length;
22     double width;
23 public:
24     Rectangle(double l,double w):
25         length(l),width(w) {}
26     double getArea() const override{
27         return length * width;
28     }
29 };
30
31 int main() {
32     Circle circle(5.0);
33     Rectangle rectangle(4.0, 6.0);
34     cout<<circle.getArea()<<endl;
35     cout<<rectangle.getArea()<<endl;
36     return 0;

```