

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	1
3.1	最長迴文子字串	1
3.2	stringstream	2
4	STL	2
4.1	priority_queue	2
4.2	deque	2
4.3	map	2
4.4	unordered_map	3
4.5	set	3
4.6	multiset	3
4.7	unordered_set	3
4.8	單調隊列	3
5	sort	4
5.1	大數排序	4
6	math	5
6.1	質數與因數	5
6.2	prime factorization	5
6.3	快速冪	5
6.4	歐拉函數	6
7	algorithm	6
7.1	basic	6
7.2	binarysearch	6
7.3	prefix sum	7
7.4	差分	7
7.5	greedy	7
7.6	floydwarshall	10
8	動態規劃	10
8.1	LCS 和 LIS	10
9	Section2	10
9.1	thm	10

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

1	int	char	int	char	int	char
2	32		64	@	96	`
3	33	!	65	A	97	a
4	34	"	66	B	98	b
5	35	#	67	C	99	c
6	36	\$	68	D	100	d
7	37	%	69	E	101	e
8	38	&	70	F	102	f

9	39	'	71	G	103	g
10	40	(72	H	104	h
11	41)	73	I	105	i
12	42	*	74	J	106	j
13	43	+	75	K	107	k
14	44	,	76	L	108	l
15	45	-	77	M	109	m
16	46	.	78	N	110	n
17	47	/	79	O	111	o
18	48	0	80	P	112	p
19	49	1	81	Q	113	q
20	50	2	82	R	114	r
21	51	3	83	S	115	s
22	52	4	84	T	116	t
23	53	5	85	U	117	u
24	54	6	86	V	118	v
25	55	7	87	W	119	w
26	56	8	88	X	120	x
27	57	9	89	Y	121	y
28	58	:	90	Z	122	z
29	59	;	91	[123	{
30	60	<	92	\	124	
31	61	=	93]	125	}
32	62	>	94	^	126	~
33	63	?	95	_		

2.2 limits

	[Type]	[size]	[range]
1	char	1	127 to -128
2	signed char	1	127 to -128
3	unsigned char	1	0 to 255
4	short	2	32767 to -32768
5	int	4	2147483647 to -2147483648
6	unsigned int	4	0 to 4294967295
7	long	4	2147483647 to -2147483648
8	unsigned long	4	0 to 18446744073709551615
9	long long	8	9223372036854775807 to -9223372036854775808
10	double	8	1.79769e+308 to 2.22507e-308
11	long double	16	1.18973e+4932 to 3.3621e-4932
12	float	4	3.40282e+38 to 1.17549e-38
13	unsigned long long	8	0 to 18446744073709551615
14	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。
2
3 讀取優先權最高的值：
4     x = pq.top();
5     pq.pop();           //讀取後刪除
6 判斷是否為空的priority_queue：
7     pq.empty()          //回傳 true
8     pq.size()           //回傳 0
9 如需改變priority_queue的優先權定義：
10    priority_queue<T> pq; //預設由大到小
11    priority_queue<T, vector<T>, greater<T>> > pq;
12                                //改成由小到大
13    priority_queue<T, vector<T>, cmp> pq; //cmp

```

4.2 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，
8     讓我們不必煩惱佇列長度的問題。
9 dq.push_back() //在 deque 的最尾端新增元素
10 dq.push_front() //在 deque 的開頭新增元素
11 dq.pop_back() //移除 deque 最尾端的元素
12 dq.pop_front() //移除 deque 最開頭的元素
13 dq.back() //取出 deque 最尾端的元素
14 dq.front() //回傳 deque 最開頭的元素
15 dq.insert()
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()
20     //刪除元素，需要使用迭代器指定刪除的元素或位置，
21     //同時也會返回指向刪除元素下一元素的迭代器。
22 dq.clear() //清空整個 deque 佇列。
23 dq.size() //檢查 deque 的尺寸
24 dq.empty() //如果 deque 佇列為空返回 1；
25     若是存在任何元素，則返回 0
26 dq.begin() //返回一個指向 deque 開頭的迭代器
27 dq.end() //指向 deque 結尾，
28     不是最後一個元素，
29     而是最後一個元素的下一個位置

```

4.3 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
12 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
16 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
21 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
27 查找
28 count(): 回傳指定元素出現的次數
29 find(): 查找一個元素
30
31 //實作範例
32 #include <bits/stdc++.h>
33 using namespace std;
34 int main(){
35     //declaration container and iterator
36     map<string, string> mp;
37     map<string, string>::iterator iter;
38     map<string, string>::reverse_iterator iter_r;
39
40     //insert element
41     mp.insert(pair<string, string>("r000",
42         "student_zero"));
43     mp["r123"] = "student_first";
44     mp["r456"] = "student_second";
45
46     //traversal
47     for(iter = mp.begin(); iter != mp.end(); iter++)
48         cout << iter->first << " " << iter->second << endl;
49     for(iter_r = mp.rbegin(); iter_r != mp.rend();
50         iter_r++)
51         cout << iter_r->first << " "
52             << iter_r->second << endl;
53
54     //find and erase the element
55     iter = mp.find("r123");
56     mp.erase(iter);

```

```

54     iter = mp.find("r123");
55     if(iter != mp.end())
56         cout<<"Find, the value is"
57             "<<iter->second<<endl;
58     else
59         cout<<"Do not Find"<<endl;
60     return 0;

```

4.4 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

4.5 set

1 set：集合，去除重複的元素，資料由小到大排序。
 2
 3 取值：使用iterator
 4 x = *st.begin();
 5 // set中的第一個元素(最小的元素)。
 6 x = *st.rbegin();
 7 // set中的最後一個元素(最大的元素)。
 8
 9 判斷是否為空的set：
 10 st.empty() 回傳true
 11 st.size() 回傳零
 12
 13 常用來搭配的member function：
 14 st.count(x);
 15 auto it = st.find(x);
 16 // binary search, O(log(N))
 17 auto it = st.lower_bound(x);
 18 // binary search, O(log(N))
 19 auto it = st.upper_bound(x);
 20 // binary search, O(log(N))

4.6 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);
 7 //會刪除所有值為 val 的元素。
 8 st.erase(st.find(val));
 9 //只刪除第一個值為 val 的元素。

4.7 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。
 5
 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 int main() {
22     unordered_set<int> myunordered_set = {3, 1};
23     myunordered_set.insert(2);
24     myunordered_set.insert(5);
25     myunordered_set.insert(4);
26     myunordered_set.insert(5);
27     myunordered_set.insert(4);
28     for (const auto &s : myunordered_set)
29         cout << s << " ";
30     cout << "\n";
31     return 0;
32 }
33
34 /*
35 output
36 4 5 2 1 3
37 */
38
39 unordered_set 刪除指定元素
40 #include <iostream>
41 #include <unordered_set>
42 int main() {
43     unordered_set<int> myunordered_set{2, 4, 6, 8};
44     myunordered_set.erase(2);
45     for (const auto &s : myunordered_set)
46         cout << s << " ";
47     cout << "\n";
48     return 0;
49 }
50 /*
51 output
52 8 6 4
53 */
54
55 清空 unordered_set 元素
56 unordered_set<int> myunordered_set;
57 myunordered_set.insert(1);
58 myunordered_set.clear();
59
60 unordered_set 判斷元素是否存在
61 unordered_set<int> myunordered_set;
62 myunordered_set.insert(2);
63 myunordered_set.insert(4);
64 myunordered_set.insert(6);
65 cout << myunordered_set.count(4) << "\n"; // 1
66 cout << myunordered_set.count(8) << "\n"; // 0
67
68 判斷 unordered_set 容器是否為空
69 #include <iostream>
70 #include <unordered_set>
71
72 int main() {
73     unordered_set<int> myunordered_set;
74     myunordered_set.clear();
75     if(myunordered_set.empty())
76         cout<<"empty\n";
77     else
78         cout<<"not empty, size is"
79             "<<myunordered_set.size()<<"\n";
80     return 0;

```

4.8 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++) {
36         while(head<=tail&&a[q[tail]]<=a[i])tail--;
37         q[++tail] = i;
38     }
39     for (int i = k; i <= n; i++) {
40         while(head<=tail&&a[q[tail]]<=a[i])tail--;
41         q[++tail] = i;
42         while (q[head] <= i - k) head++;
43         cout<<a[q[head]]<<" ";
44     }
45 }
46
47 int main() {
48     cin>>n>>k; //每k個連續的數
49     for (int i = 1; i <= n; i++) cin>>a[i];
50     getmin();
51     cout<<"\n";
52     getmax();
53     cout<<"\n";
54     return 0;
55 }
56
57 //寫法2
58 #include <iostream>
59 #include <cstring>
60 #include <deque>
61 using namespace std;
62 int a[1000005];
63
64 int main() {
65     ios_base::sync_with_stdio(0);
66     int n, k;
67     while(cin>>n>>k) {
68         for(int i=0; i<n; i++) cin >> a[i];
69         deque<int> dq;
70         for(int i=0; i<n; i++){
71             while(dq.size() && dq.front()<=i-k)
72                 dq.pop_front();
73             while(dq.size() && a[dq.back()]>a[i])
74                 dq.pop_back();
75             dq.push_back(i);
76             if(i==k-1) cout<<a[dq.front()];
77             if(i>k-1) cout<<' '<<a[dq.front()];
78         }
79         if(k>n) cout<<a[dq.front()];
80         cout<<"\n";
81         while(dq.size()) dq.pop_back();
82         for(int i=0; i<n; i++){
83             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 }
94
95 example 2
96
97 一個含有 n 項的數列，求出每一項前的 m
98 個數到它這個區間內的最小值。
99 若前面的數不足 m 項則從第 1
100 個數開始，若前面沒有數則輸出 0
101
102 #include<bits/stdc++.h>
103 using namespace std;
104 #define re register int
105 #define INF 0x3f3f3f3f
106 #define ll long long
107 #define maxn 2000009
108 #define maxm
109 inline ll read() {
110     ll x=0,f=1;
111     char ch=getchar();
112     while(ch<'0' || ch>'9'){
113         if(ch=='-') f=-1;
114         ch=getchar();
115     }
116     while(ch>='0' && ch<='9'){
117         x=(x<<1)+(x<<3)+(ll)(ch-'0');
118         ch=getchar();
119     }
120     return x*f;
121 }
122 int n,m,k,tot,head,tail;
123 int a[maxn],q[maxn];
124 int main() {
125     n=read(), m=read();
126     for(int i=1;i<=n;i++) a[i]=read();
127     head=1,tail=0; //起始位置為1
128     //因為插入是q[++tail]所以要初始化為0
129     for(int i=1;i<=n;i++)
130     {
131         //每次隊首的元素就是當前的答案
132         cout<<a[q[head]]<<endl;
133         while(i-q[head]+1>m&&head<=tail) //維護隊首
134             head++;
135         while(a[i]<a[q[tail]]&&head<=tail) //維護隊尾
136             tail--;
137         q[++tail]=i;
138     }
139     return 0;
140 }

```

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

```

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 歐拉篩O(n)
13 #define MAXN 47000 // sqrt(2^31) = 46,340...
14 bool isPrime[MAXN];
15 int prime[MAXN];
16 int primeSize = 0;
17 void getPrimes(){
18     memset(isPrime, true, sizeof(isPrime));
19     isPrime[0] = isPrime[1] = false;
20     for (int i = 2; i < MAXN; i++){
21         if (isPrime[i]) prime[primeSize++] = i;
22         for (int j = 0; j < primeSize && i * prime[j]
23             <= MAXN; ++j){
24             isPrime[i * prime[j]] = false;
25             if (i % prime[j] == 0) break;
26         }
27     }
28
29 因數
30
31 最大公因數 O(log(min(a,b)))
32 int GCD(int a, int b)
33 {
34     if (b == 0) return a;
35     return GCD(b, a % b);
36 }
37
38 質因數分解
39
40 void primeFactorization(int n){
41     for(int i=0;i<(int)p.size();++i){
42         if(p[i] * p[i] > n) break;
43         if(n % p[i]) continue;
44         cout << p[i] << ' ';
45         while(n % p[i] == 0) n /= p[i];
46     }
47     if(n!=1) cout<<n<< ' ';
48     cout<<'\n';
49 }
50
51 歌德巴赫猜想
52 solution : 把偶數 N (6≤N≤10^6) 寫成兩個質數的和。
53 #include <iostream>
54 #include <cstdio>
55 using namespace std;
56 #define N 20000000
57 int ox[N], p[N], pr;
58 void PrimeTable(){
59     ox[0] = ox[1] = 1;
60     pr = 0;
61     for (int i = 2; i < N; i++){

```

```

63         if (!ox[i]) p[pr++] = i;
64         for (int j = 0; i*p[j]<N&&j < pr; j++)
65             ox[i*p[j]] = 1;
66     }
67 }
68
69 int main(){
70     PrimeTable();
71     int n;
72     while (cin>>n,n){
73         int x;
74         for (x = 1;; x += 2)
75             if (!ox[x] && !ox[n - x])break;
76         printf("%d = %d + %d\n", n, x, n - x);
77     }
78 }
79 problem : 給定整數 N，求 N
80     最少可以拆成多少個質數的和。
81 如果 N 是質數，則答案為 1。
82 如果 N 是偶數(不包含2)，則答案為 2 (強歌德巴赫猜想)。
83 如果 N 是奇數且 N-2 是質數，則答案為 2 (2+質數)。
84 其他狀況答案為 3 (弱歌德巴赫猜想)。
85 #pragma GCC optimize("O2")
86 #include <bits/stdc++.h>
87 using namespace std;
88 #define FOR(i, L, R) for(int i=L;i<(int)R;++i)
89 #define FORD(i, L, R) for(int i=L;i>(int)R;--i)
90 #define IOS
91 cin.tie(nullptr);
92 cout.tie(nullptr);
93 ios_base::sync_with_stdio(false);
94
95 bool isPrime(int n){
96     FOR(i, 2, n){
97         if (i * i > n)
98             return true;
99         if (n % i == 0)
100             return false;
101     }
102     return true;
103 }
104
105 int main(){
106     IOS;
107     int n;
108     cin >> n;
109     if(isPrime(n)) cout << "1\n";
110     else if(n%2==0||isPrime(n-2)) cout<<"2\n";
111     else cout << "3\n";

```

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
18 int main(){
19     while(cin >> n)
20         cout << phi() << endl;
21 }

```

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，
11     如果不存在，則回傳 last。
12 lower_bound(first, last, val)
13 upper_bound：尋找第一個大於 x 的元素位置，
14     如果不存在，則回傳 last。
15 upper_bound(first, last, val)
16 next_permutation：將序列順序轉換成下一個字典序，
17     如果存在回傳 true，反之回傳 false。
18 next_permutation(first, last)
19 prev_permutation：將序列順序轉換成上一個字典序，
20     如果存在回傳 true，反之回傳 false。
21 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。
8
9 Scarecrow
10 //problem
11 有一個 N×1 的稻田，有些稻田現在有種植作物，
12 為了避免被動物破壞，需要放置稻草人，
13 稻草人可以保護該塊稻田和左右兩塊稻田，
14 請問最少需要多少稻草人才能保護所有稻田？
15
16 //solution
17 從左到右掃描稻田，如果第 i 塊稻田有作物，
18 就把稻草人放到第 i+1 塊稻田，
19 這樣能保護第 i, i+1, i+2 塊稻田，
20 接著從第 i+3 塊稻田繼續掃描。
21
22 //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
38 霍夫曼樹的變形題
39 //problem
40 給定 N 個數，每次將兩個數 a, b 合併成 a+b，
41 只到最後只剩一個數，合併成本為兩數和，
42 問最小合併成本為多少。
43
44 //solution
45 每次將最小的兩數合併起來。
46
47 //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 }
72
73 刪數字問題
74 //problem
75 給定一個數字 N (≤10^100)，需要刪除 K 個數字，
76 請問刪除 K 個數字後最小的數字為何？
77
78 //solution
79 刪除滿足第 i 位數大於第 i+1 位數的最左邊第 i 位數，
80 扣除高位數的影響較扣除低位數的大。
81
82 //code
83 int main()
84 {
85     string s;
86     int k;
87     cin >> s >> k;
88     for (int i = 0; i < k; ++i){
89         if ((int)s.size() == 0) break;
90         int pos = (int)s.size() - 1;
91         for (int j = 0; j < (int)s.size() - 1; ++j){
92             if (s[j] > s[j + 1]){
93                 pos = j;
94                 break;
95             }
96         }
97         s.erase(pos, 1);
98     }
99     while ((int)s.size() > 0 && s[0] == '0')
100         s.erase(0, 1);
101     if ((int)s.size()) cout << s << '\n';
102     else cout << 0 << '\n';
103 }
104
105 區間覆蓋長度
106 //problem
107 給定 n 條線段區間為 [Li, Ri]，

```

108 請問這些線段的覆蓋所覆蓋的長度？
 109
 110 *//solution*
 111 先將所有區間依照左界由小到大排序，
 112 左界相同依照右界由小到大排序，
 113 用一個變數 R 紀錄目前最大可以覆蓋到的右界。
 114 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。

```
115
116 //code
117 struct Line
118 {
119     int L, R;
120     bool operator<(const Line &rhs) const
121     {
122         if (L != rhs.L) return L < rhs.L;
123         return R < rhs.R;
124     }
125 };
126
127 int main(){
128     int n;
129     Line a[10005];
130     while (cin >> n){
131         for (int i = 0; i < n; i++)
132             cin >> a[i].L >> a[i].R;
133         sort(a, a + n);
134         int ans = 0, L = a[0].L, R = a[0].R;
135         for (int i = 1; i < n; i++){
136             if (a[i].L < R) R = max(R, a[i].R);
137             else{
138                 ans += R - L;
139                 L = a[i].L;
140                 R = a[i].R;
141             }
142         }
143         cout << ans + (R - L) << '\n';
144     }
145 }
```

148 最小區間覆蓋長度

149 *//problem*
 150 給定 n 條線段區間為 $[Li, Ri]$ ，
 151 請問最少要選幾個區間才能完全覆蓋 $[0, S]$ ？

152 *//solution*
 153 先將所有區間依照左界由小到大排序，
 154 對於當前區間 $[Li, Ri]$ ，要從左界 $> Ri$ 的所有區間中，
 155 找到有著最大的右界的區間，連接當前區間。

156 *//problem*
 157 長度 n 的直線中有數個加熱器，
 158 在 x 的加熱器可以讓 $[x-r, x+r]$ 內的物品加熱，
 159 問最少要幾個加熱器可以把 $[0, n]$ 的範圍加熱。
 160
 161 *//solution*
 162 對於最左邊沒加熱的點 a，選擇最遠可以加熱 a 的加熱器，
 163 更新已加熱範圍，重複上述動作繼續尋找加熱器。

```
164 //code
165 int main(){
166     int n, r;
167     int a[1005];
168     cin >> n >> r;
169     for (int i=1; i<=n; ++i) cin>>a[i];
170     int i = 1, ans = 0;
171     while (i <= n){
172         int R=min(i+r-1, n), L=max(i-r+1, 0)
173         int nextR=-1;
174         for (int j = R; j >= L; --j){
175             if (a[j]){
176                 nextR = j;
177                 break;
178             }
179         }
180     }
181     if (nextR == -1){
```

```
182         ans = -1;
183         break;
184     }
185     ++ans;
186     i = nextR + r;
187 }
188 cout << ans << '\n';
189 }
190
191
192
193
194 最多不重疊區間
195 //problem
196 給你 n 條線段區間為  $[Li, Ri]$ ，
197 請問最多可以選擇幾條不重疊的線段(頭尾可相連)?
198
199 //solution
200 依照右界由小到大排序，
201 每次取到一個不重疊的線段，答案 +1。
202
203 //code
204 struct Line
205 {
206     int L, R;
207     bool operator<(const Line &rhs) const {
208         return R < rhs.R;
209     }
210 };
211
212 int main(){
213     int t;
214     cin >> t;
215     Line a[30];
216     while (t--){
217         int n = 0;
218         while (cin>>a[n].L>>a[n].R, a[n].L||a[n].R)
219             ++n;
220         sort(a, a + n);
221         int ans = 1, R = a[0].R;
222         for (int i = 1; i < n; i++){
223             if (a[i].L >= R){
224                 ++ans;
225                 R = a[i].R;
226             }
227         }
228         cout << ans << '\n';
229     }
230 }
```

231 區間選點問題

232 *//problem*
 233 給你 n 條線段區間為 $[Li, Ri]$ ，
 234 請問至少要取幾個點才能讓每個區間至少包含一個點？

235 *//solution*
 236 將區間依照右界由小到大排序，R=第一個區間的右界，
 237 遍歷所有區段，如果當前區間左界 $> R$ ，
 238 代表必須多選一個點 ($ans+1$)，並將 R=當前區間右界。

239 *//problem*
 240 給定 N 個座標，要在 x 軸找到最小的點，
 241 讓每個座標至少和一個點距離 $\leq D$ 。

242 *//solution*
 243 以每個點 (xi, yi) 為圓心半徑為 D 的圓 C，
 244 求出 C 和 x 軸的交點 Li, Ri ，題目轉變成區間選點問題。

```
245 //code
246 struct Line
247 {
248     int L, R;
249     bool operator<(const Line &rhs) const {
250         return R < rhs.R;
251     }
252 };
253
```



```

260 int main(){
261     int t;
262     cin >> t;
263     Line a[30];
264     while (t--){
265         int n = 0;
266         while (cin>>a[n].L>>a[n].R, a[n].L||a[n].R)
267             ++n;
268         sort(a, a + n);
269         int ans = 1, R = a[0].R;
270         for (int i = 1; i < n; i++){
271             if (a[i].L >= R){
272                 ++ans;
273                 R = a[i].R;
274             }
275         }
276         cout << ans << '\n';
277     }
278 }

```

最小化最大延遲問題

給定 N 項工作，每項工作的需要處理時長為 T_i ，期限是 D_i ，第 i 項工作延遲的時間為 $L_i = \max(0, F_i - D_i)$ ，原本 F_i 為第 i 項工作的完成時間，求一種工作排序使 $\max L_i$ 最小。

//solution

按照到期時間從早到晚處理。

//code

```

291 struct Work
292 {
293     int t, d;
294     bool operator<(const Work &rhs) const {
295         return d < rhs.d;
296     }
297 };
298
299 int main(){
300     int n;
301     Work a[10000];
302     cin >> n;
303     for (int i = 0; i < n; ++i)
304         cin >> a[i].t >> a[i].d;
305     sort(a, a + n);
306     int maxL = 0, sumT = 0;
307     for (int i = 0; i < n; ++i){
308         sumT += a[i].t;
309         maxL = max(maxL, sumT - a[i].d);
310     }
311     cout << maxL << '\n';
312 }

```

最少延遲數量問題

//problem

給定 N 個工作，每個工作的需要處理時長為 T_i ，期限是 D_i ，求一種工作排序使得逾期工作數量最小。

//solution

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，依序放入工作列表中，如果發現有工作預期，就從目前選擇的工作中，移除耗時最長的工作。

上述方法為 Moore-Hodgson's Algorithm。

//problem

給定烏龜的重量和可承受重量，問最多可以疊幾隻烏龜？

和最少延遲數量問題是相同的問題，只要將題敘做轉換。

工作處理時長 → 烏龜重量

工作期限 → 烏龜可承受重量

多少工作不延期 → 可以疊幾隻烏龜

//code

```

335 struct Work{
336     int t, d;
337     bool operator<(const Work &rhs) const {
338         return d < rhs.d;
339     }
340 };
341
342 int main(){
343     int n = 0;
344     Work a[10000];
345     priority_queue<int> pq;
346     while (cin >> a[n].t >> a[n].d)
347         ++n;
348     sort(a, a + n);
349     int sumT = 0, ans = n;
350     for (int i = 0; i < n; ++i){
351         pq.push(a[i].t);
352         sumT += a[i].t;
353         if (a[i].d < sumT){
354             int x = pq.top();
355             pq.pop();
356             sumT -= x;
357             --ans;
358         }
359     }
360     cout << ans << '\n';
361 }

```

任務調度問題

//problem

給定 N 項工作，每項工作的需要處理時長為 T_i ，期限是 D_i ，如果第 i 項工作延遲需要受到 p_i 單位懲罰，請問最少會受到多少單位懲罰。

//solution

依照懲罰由大到小排序，

每項工作依序嘗試可不可以放在 $D_i - T_i + 1, D_i - T_i, \dots, 1, 0$ ，如果有空閒就放進去，否則延後執行。

//problem

給定 N 項工作，每項工作的需要處理時長為 T_i ，期限是 D_i ，如果第 i 項工作在期限內完成會獲得 a_i 單位獎勵，請問最多會獲得多少單位獎勵。

//solution

和上題相似，這題變成依照獎勵由大到小排序。

//code

```

386 struct Work
387 {
388     int d, p;
389     bool operator<(const Work &rhs) const {
390         return p > rhs.p;
391     }
392 };
393
394 int main(){
395     int n;
396     Work a[100005];
397     bitset<100005> ok;
398     while (cin >> n){
399         ok.reset();
400         for (int i = 0; i < n; ++i)
401             cin >> a[i].d >> a[i].p;
402         sort(a, a + n);
403         int ans = 0;
404         for (int i = 0; i < n; ++i){
405             int j = a[i].d;
406             while (j--){
407                 if (!ok[j]){
408                     ans += a[i].p;
409                     ok[j] = true;

```

```

410         break;
411     }
412 }
413 cout << ans << '\n';
414 }
415 }
416
417 多機調度問題
418 //problem
419 給定 N 項工作，每項工作的需要處理時長為 Ti，
420 有 M 台機器可執行多項工作，但不能將工作拆分，
421 最快可以在什麼時候完成所有工作？
422
423 //solution
424 將工作由大到小排序，每項工作交給最快空閒的機器。
425
426 //code
427 int main(){
428     int n, m;
429     int a[10000];
430     cin >> n >> m;
431     for (int i = 0; i < n; ++i)
432         cin >> a[i];
433     sort(a, a + n, greater<int>());
434     int ans = 0;
435     priority_queue<int, vector<int>, greater<int>> pq;
436     for (int i = 0; i < m && i < n; ++i){
437         ans = max(ans, a[i]);
438         pq.push(a[i]);
439     }
440     for (int i = m; i < n; ++i){
441         int x = pq.top();
442         pq.pop();
443         x += a[i];
444         ans = max(ans, x);
445         pq.push(x);
446     }
447     cout << ans << '\n';
448 }

```

7.6 floydwarshall

```

1 int w[n][n];
2 int d[n][n];
3 int medium[n][n];
4 // 由 i 點到 j 點的路徑，其中繼點為 medium[i][j]。
5
6 void floyd_warshall(){
7     for (int i=0; i<n; i++){
8         for (int j=0; j<n; j++){
9             d[i][j] = w[i][j];
10            medium[i][j] = -1;
11            // 預設為沒有中繼點
12        }
13    }
14    for(int i=0; i<n; i++) d[i][i]=0;
15    for(int k=0; k<n; k++){
16        for(int i=0; i<n; i++){
17            for(int j=0; j<n; j++){
18                if(d[i][k]+d[k][j]<d[i][j]){
19                    d[i][j]=d[i][k]+d[k][j];
20                    medium[i][j]=k;
21                    // 由 i 點走到 j 點經過了 k 點
22                }
23            }
24        }
25    }
26    // 這支函式並不會印出起點和終點，必須另行印出。
27    void find_path(int s, int t){ // 印出最短路徑
28        if (medium[s][t] == -1) return; // 沒有中繼點就結束
29        find_path(s, medium[s][t]); // 前半段最短路徑
30        cout << medium[s][t]; // 中繼點
31        find_path(medium[s][t], t); // 後半段最短路徑
32    }
33 }

```

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 Section2

9.1 thm

- 中文測試
- $$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$