

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
4	STL	2
4.1	priority_queue . . . . .	2
4.2	queue . . . . .	2
4.3	deque . . . . .	2
4.4	map . . . . .	2
4.5	unordered_map . . . . .	3
4.6	set . . . . .	3
4.7	multiset . . . . .	3
4.8	unordered_set . . . . .	3
4.9	單調隊列 . . . . .	4
5	sort	5
5.1	大數排序 . . . . .	5
5.2	bubble sort . . . . .	5
6	math	5
6.1	質數與因數 . . . . .	5
6.2	prime factorization . . . . .	6
6.3	快速冪 . . . . .	6
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	hw6 . . . . .	13

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
1	32		64	@	96	`
2	33	!	65	A	97	a
3	34	"	66	B	98	b
4	35	#	67	C	99	c
5	36	\$	68	D	100	d
6	37	%	69	E	101	e
7	38	&	70	F	102	f
8	39	'	71	G	103	g
9	40	(	72	H	104	h
10	41	)	73	I	105	i
11	42	*	74	J	106	j
12	43	+	75	K	107	k
13	44	,	76	L	108	l
14	45	-	77	M	109	m
15	46	.	78	N	110	n
16	47	/	79	O	111	o
17	48	0	80	P	112	p
18	49	1	81	Q	113	q
19	50	2	82	R	114	r
20	51	3	83	S	115	s
21	52	4	84	T	116	t
22	53	5	85	U	117	u
23	54	6	86	V	118	v
24	55	7	87	W	119	w
25	56	8	88	X	120	x
26	57	9	89	Y	121	y
27	58	:	90	Z	122	z
28	59	;	91	[	123	{
29	60	<	92	\	124	
30	61	=	93	]	125	}
31	62	>	94	^	126	~
32	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	
10		9223372036854775807 to -9223372036854775808	
11	double	8	1.79769e+308 to 2.22507e-308
12	long double	16	1.18973e+4932 to 3.3621e-4932
13	float	4	3.40282e+38 to 1.17549e-38
14	unsigned long long	8	0 to 18446744073709551615
15	string	32	
16			

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(1 - i >= 0 && r + i < n && T(1 - 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 }

```

## 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: " <<
            q.front() << endl;
37     }
38 }

```

### 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，
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 dq.insert(position, n, val)
17     position: 插入元素的 index 值
18     n: 元素插入次數
19     val: 插入的元素值
20 dq.erase() //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，
    //不是最後一個元素，
    //而是最後一個元素的下一個位置

```

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

```

```

77 while(n--){
78     cin>>x;
79     ++mp[x];
80 }
81 for(auto i:mp) cout<<i.first<<" "<<i.second<<endl;
82 }

```

## 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
3 宣告：
4     set <int> st;
5
6 把元素 x 加進 set：
7     st.insert(x);
8
9 檢查元素 x 是否存在 set 中：
10    st.count(x);
11
12 刪除元素 x：
13    st.erase(x); // 可傳入值或iterator
14
15 清空集合中的所有元素：
16    st.clear();
17
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)。
23
24 判斷是否為空的set：
25    st.empty() 回傳true
26    st.size() 回傳零
27
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。
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
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++) {
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     getmax();
52     cout << '\n';
53     return 0;
54 }
55
56 //寫法2
57 #include <iostream>
58 #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 inline ll read() {
108     ll x=0,f=1;
109     char ch=getchar();
110     while(ch<'0' || ch>'9'){
111         if(ch=='-') f=-1;
112         ch=getchar();
113     }
114     while(ch>='0' && ch<='9'){
115         x=(x<<1)+(x<<3)+(ll)(ch-'0');
116         ch=getchar();
117     }
118     return x*f;
119 }
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;

```

```

131         while(i-q[head]+1>m&&head<=tail) //維護隊首
132             head++;
133         while(a[i]<a[q[tail]]&&head<=tail) //維護隊尾
134             tail--;
135         q[++tail]=i;
136     }
137     return 0;
138 }

```

## 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 一般篩法  $O(N \log \log N)$ 
4 vector<int> p;
5 bitset<MAXN> is_notp;
6 void PrimeTable(int n)
7 {
8     is_notp.reset();
9     is_notp[0] = is_notp[1] = 1;
10    for (int i = 2; i <= n; i++)
11    {
12        if (is_notp[i])
13            continue;
14        p.push_back(i);
15        for (int j = i * i; j <= n; j += i)

```

```

16     {
17         is_notp[j] = 1;
18     }
19 }
20 }
21
22 線性篩法 O(N)
23 vector<int> p;
24 bitset<MAXN> is_notp;
25 void PrimeTable(int n)
26 {
27     is_notp.reset();
28     is_notp[0] = is_notp[1] = 1;
29     for (int i = 2; i <= n; ++i)
30     {
31         if (!is_notp[i])
32             p.push_back(i);
33         for (int j = 0; j < (int)p.size(); ++j)
34         {
35             if (i * p[j] > n)
36                 break;
37             is_notp[i * p[j]] = 1;
38             if (i % p[j] == 0)
39                 break;
40         }
41     }
42 }
43
44 因數
45
46 最大公因數 O(log(min(a,b)))
47 int GCD(int a, int b)
48 {
49     if (b == 0) return a;
50     return GCD(b, a % b);
51 }
52
53 質因數分解
54
55 void primeFactorization(int n)
56 {
57     for (int i = 0; i < (int)p.size(); ++i)
58     {
59         if (p[i] * p[i] > n)
60             break;
61         if (n % p[i])
62             continue;
63         cout << p[i] << ' ';
64         while (n % p[i] == 0)
65             n /= p[i];
66     }
67     if (n != 1)
68         cout << n << ' ';
69     cout << '\n';
70 }
71
72 歌德巴赫猜想
73 solution : 把偶數 N ( $6 \leq N \leq 10^6$ ) 寫成兩個質數的和。
74 #include <iostream>
75 #include <cstdio>
76 using namespace std;
77 #define N 2000000
78 int ox[N], p[N], pr;
79
80 void PrimeTable(){
81     ox[0] = ox[1] = 1;
82     pr = 0;
83     for (int i = 2; i < N; i++){
84         if (!ox[i]) p[pr++] = i;
85         for (int j = 0; i*p[j]<N&&j < pr; j++)
86             ox[i*p[j]] = 1;
87     }
88 }
89
90 int main(){
91     PrimeTable();
92     int n;

```

```

93     while (cin>>n,n){
94         int x;
95         for (x = 1;; x += 2)
96             if (!ox[x] && !ox[n - x])break;
97         printf("%d = %d + %d\n", n, x, n - x);
98     }
99 }
100 problem : 給定整數 N，求 N
101         最少可以拆成多少個質數的和。
102 如果 N 是質數，則答案為 1。
103 如果 N 是偶數(不包含2)，則答案為 2 (強歌德巴赫猜想)。
104 如果 N 是奇數且 N-2 是質數，則答案為 2 (2+質數)。
105 其他狀況答案為 3 (弱歌德巴赫猜想)。
106 #pragma GCC optimize("O2")
107 #include <bits/stdc++.h>
108 using namespace std;
109 #define FOR(i, L, R) for(int i=L;i<(int)R;++i)
110 #define FORD(i, L, R) for(int i=L;i>(int)R;--i)
111 #define IOS
112     cin.tie(nullptr);
113     cout.tie(nullptr);
114     ios_base::sync_with_stdio(false);
115
116 bool isPrime(int n){
117     FOR(i, 2, n){
118         if (i * i > n)
119             return true;
120         if (n % i == 0)
121             return false;
122     }
123     return true;
124 }
125
126 int main(){
127     IOS;
128     int n;
129     cin >> n;
130     if(isPrime(n)) cout << "1\n";
131     else if(n%2==0||isPrime(n-2)) cout<<"2\n";
132     else cout << "3\n";
133 }

```

## 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 }

```

## 7 algorithm

### 7.1 basic

```

1  min：取最小值。
2  min(a, b)
3  min(list)
4  max：取最大值。
5  max(a, b)
6  max(list)
7  min_element：找尋最小元素
8  min_element(first, last)
9  max_element：找尋最大元素
10 max_element(first, last)
11 sort：排序，預設由小排到大。
12 sort(first, last)
13 sort(first, last, comp)：可自行定義比較運算子 Comp。
14 find：尋找元素。
15 find(first, last, val)
16 lower_bound：尋找第一個小於 x
   的元素位置，如果不存在，則回傳 last。
17 lower_bound(first, last, val)
18 upper_bound：尋找第一個大於 x
   的元素位置，如果不存在，則回傳 last。
19 upper_bound(first, last, val)
20 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。
21 next_permutation(first, last)
22 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。
23 prev_permutation(first, last)
24
25

```

### 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 }

```

霍夫曼樹的變形題

```

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

```

```

42 //solution
43 每次將最小的兩數合併起來。

```

```

44 //code
45 #include <bits/stdc++.h>
46 using namespace std;
47 int main()
48 {
49     int n, x;
50     while (cin >> n, n){
51         priority_queue<int, vector<int>, greater<int>>
52             q;
53         while (n--){
54             cin >> x;

```

```

55             q.push(x);
56         }
57         long long ans = 0;
58         while (q.size() > 1){
59             x = q.top();
60             q.pop();
61             x += q.top();
62             q.pop();
63             q.push(x);
64             ans += x;
65         }
66         cout << ans << endl;
67     }
68 }
69 }
70 }
71

```

Commando War

```

72 //problem
73 有 n 個部下，每個部次要花 Bi 分鐘交待任務，
74 再花 Ji 分鐘執行任務，一次只能對一位部下交代任務，
75 但可以多人同時執行任務，問最少要花多少時間完成任務。

```

```

76 //solution
77 執行時間長的人先交代任務
78
79 //code
80 #include <bits/stdc++.h>
81 using namespace std;
82 struct Data{
83     int b, j;
84     bool operator<(const Data &rhs) const {
85         return j > rhs.j;
86     }
87 };
88
89 int main(){
90     int n, ti = 0;
91     Data a[1005];
92     while (cin >> n, n){
93         for (int i = 0; i < n; ++i)
94             cin >> a[i].b >> a[i].j;
95         sort(a, a + n);
96         int ans = 0, sum = 0;
97         for (int i = 0; i < n; ++i){
98             sum += a[i].b;
99             ans = max(ans, sum + a[i].j);
100         }
101         cout << "Case " << ++ti << ": " << ans << '\n';
102     }
103 }
104 }
105 }
106

```

刪數字問題

```

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

```

```

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

```

```

113 //code
114 int main()
115 {
116     string s;
117     int k;
118     cin >> s >> k;
119     for (int i = 0; i < k; ++i){
120         if ((int)s.size() == 0) break;
121         int pos = (int)s.size() - 1;
122         for (int j = 0; j < (int)s.size() - 1; ++j){
123             if (s[j] > s[j + 1]){
124                 pos = j;
125                 break;
126             }
127         }
128         s.erase(pos, 1);
129     }
130 }

```



```

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
140 //problem
141 給定 n 條線段區間為 [Li,Ri]，
142 請問這些線段的覆蓋所覆蓋的長度？
143
144 //solution
145 先將所有區間依照左界由小到大排序，
146 左界相同依照右界由小到大排序，
147 用一個變數 R 紀錄目前最大可以覆蓋到的右界。
148 如果目前區間左界 ≤ R，代表該區間可以和前面的線段合併。
149
150 //code
151 struct Line
152 {
153     int L, R;
154     bool operator< (const Line &rhs) const
155     {
156         if (L != rhs.L) return L < rhs.L;
157         return R < rhs.R;
158     }
159 };
160
161 int main(){
162     int n;
163     Line a[10005];
164     while (cin >> n){
165         for (int i = 0; i < n; i++)
166             cin >> a[i].L >> a[i].R;
167         sort(a, a + n);
168         int ans = 0, L = a[0].L, R = a[0].R;
169         for (int i = 1; i < n; i++){
170             if (a[i].L < R) R = max(R, a[i].R);
171             else{
172                 ans += R - L;
173                 L = a[i].L;
174                 R = a[i].R;
175             }
176         }
177         cout << ans + (R - L) << '\n';
178     }
179 }
180
181 最小區間覆蓋長度
182 //problem
183 給定 n 條線段區間為 [Li,Ri]，
184 請問最少要選幾個區間才能完全覆蓋 [0,S]？
185
186 //solution
187 先將所有區間依照左界由小到大排序，
188 對於當前區間 [Li,Ri]，要從左界 > Ri 的所有區間中，
189 找到有著最大的右界的區間，連接當前區間。
190
191 //problem
192 長度 n 的直線中有數個加熱器，
193 在 x 的加熱器可以讓 [x-r, x+r] 內的物品加熱，
194 問最少要幾個加熱器可以把 [0,n] 的範圍加熱。
195
196 //solution
197 對於最左邊沒加熱的點a，選擇最遠可以加熱a的加熱器，
198 更新已加熱範圍，重複上述動作繼續尋找加熱器。
199
200 //code
201 int main(){
202     int n, r;
203     int a[1005];
204     cin >> n >> r;
205     for (int i=1; i<=n; ++i) cin>>a[i];

```

```

208 int i = 1, ans = 0;
209 while (i <= n){
210     int R=min(i+r-1, n), L=max(i-r+1, 0)
211     int nextR=-1;
212     for (int j = R; j >= L; --j){
213         if (a[j]){
214             nextR = j;
215             break;
216         }
217     }
218     if (nextR == -1){
219         ans = -1;
220         break;
221     }
222     ++ans;
223     i = nextR + r;
224 }
225 cout << ans << '\n';
226 }
227
228 最多不重疊區間
229 //problem
230 給你 n 條線段區間為 [Li,Ri]，
231 請問最多可以選擇幾條不重疊的線段(頭尾可相連)？
232
233 //solution
234 依照右界由小到大排序，
235 每次取到一個不重疊的線段，答案 +1。
236
237 //code
238 struct Line
239 {
240     int L, R;
241     bool operator< (const Line &rhs) const {
242         return R < rhs.R;
243     }
244 };
245
246 int main(){
247     int t;
248     cin >> t;
249     Line a[30];
250     while (t--){
251         int n = 0;
252         while (cin>>a[n].L>>a[n].R, a[n].L||a[n].R)
253             ++n;
254         sort(a, a + n);
255         int ans = 1, R = a[0].R;
256         for (int i = 1; i < n; i++){
257             if (a[i].L >= R){
258                 ++ans;
259                 R = a[i].R;
260             }
261         }
262         cout << ans << '\n';
263     }
264 }
265
266 區間選點問題
267 //problem
268 給你 n 條線段區間為 [Li,Ri]，
269 請問至少要取幾個點才能讓每個區間至少包含一個點？
270
271 //solution
272 將區間依照右界由小到大排序，R=第一個區間的右界，
273 遍歷所有區段，如果當前區間左界>R，
274 代表必須多選一個點 (ans+=1)，並將 R=當前區間右界。
275
276 //problem
277 給定 N 個座標，要在 x 軸找到最小的點，
278 讓每個座標至少和一個點距離 ≤ D。
279
280 //solution
281 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  $L_i, R_i$ ，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 }
```

316 最小化最大延遲問題

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

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 int main(){
335     int n;
336     Work a[10000];
337     cin >> n;
338     for (int i = 0; i < n; ++i)
339         cin >> a[i].t >> a[i].d;
340     sort(a, a + n);
341     int maxL = 0, sumT = 0;
342     for (int i = 0; i < n; ++i){
343         sumT += a[i].t;
344         maxL = max(maxL, sumT - a[i].d);
345     }
346     cout << maxL << '\n';
347 }
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  $T_i$ ，  
354 期限是  $D_i$ ，求一種工作排序使得逾期工作數量最小。

355 //solution  
356 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  
357 依序放入工作列表中，如果發現有工作預期，  
358 就從目前選擇的工作中，移除耗時最長的工作。

360 上述方法為 Moore-Hodgson's Algorithm。

361 //problem  
362 給定烏龜的重量和可承受重量，問最多可以疊幾隻烏龜？  
363 和最少延遲數量問題是相同的問題，只要將題敘做轉換。  
364 工作處理時長 → 烏龜重量  
365 工作期限 → 烏龜可承受重量  
366 多少工作不延期 → 可以疊幾隻烏龜

```
367
368 //code
369 struct Work{
370     int t, d;
371     bool operator< (const Work &rhs) const {
372         return d < rhs.d;
373     }
374 };
375
376 int main(){
377     int n = 0;
378     Work a[10000];
379     priority_queue<int> pq;
380     while (cin >> a[n].t >> a[n].d)
381         ++n;
382     sort(a, a + n);
383     int sumT = 0, ans = n;
384     for (int i = 0; i < n; ++i){
385         pq.push(a[i].t);
386         sumT += a[i].t;
387         if (a[i].d < sumT){
388             int x = pq.top();
389             pq.pop();
390             sumT -= x;
391             --ans;
392         }
393     }
394     cout << ans << '\n';
395 }
396
397
398
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  $T_i$ ，  
404 期限是  $D_i$ ，如果第 i 項工作延遲需要受到  $p_i$  單位懲罰，  
405 請問最少會受到多少單位懲罰。

406 //solution  
407 依照懲罰由大到小排序，  
408 每項工作依序嘗試可不可以放在  $D_i - T_i + 1, D_i - T_i, \dots, 1, 0$ ，  
409 如果有空閒就放進去，否則延後執行。

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

415 //solution  
416 和上題相似，這題變成依照獎勵由大到小排序。

```
417 //code
418 struct Work
419 {
420     int d, p;
421     bool operator< (const Work &rhs) const {
422         return p > rhs.p;
423     }
424 };
425
426 int main(){
427     int n;
428     Work a[100005];
429     bitset<100005> ok;
430     while (cin >> n){
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```

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 多機調度問題
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，

```

越早出現的數字要越小

5. 如果有數字在 B 裡面而不在 A 裡面，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         deleteList(root->next);
59         delete root;
60         return;
61     }
62 };
63
64 class TreeProblem {
65     int null = INT_MIN;
66     vector<int> nums = {}, result;
67 }

```

```

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(
72             nums[mid],
73             buildBinaryTreeUsingDFS(left,mid-1),
74             buildBinaryTreeUsingDFS(mid+1,right));
75         return current;
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
86                     TreeNode(nums[idx]);
87                 q.front()->left = left;
88                 q.push(left);
89             }
90             idx++;
91             if((idx < nums.size()) && (nums[idx] !=
                null)) {
92                 TreeNode* right = new
93                     TreeNode(nums[idx]);
94                 q.front()->right = right;
95                 q.push(right);
96             }
97             idx++;
98             q.pop();
99         }
100         return root;
101     }
102
103     Node* buildNaryTree() {
104         int idx = 2;
105         Node *root = new Node(nums.front());
106         queue<Node*> q;
107         q.push(root);
108         while(idx < nums.size()) {
109             while((idx < nums.size()) && (nums[idx]
                != null)) {
110                 Node *current = new Node(nums[idx++]);
111                 q.front()->children.push_back(current);
112                 q.push(current);
113             }
114             idx++;
115             q.pop();
116         }
117         return root;
118     }
119
120     void deleteBinaryTree(TreeNode* root) {
121         if(root->left != NULL)
122             deleteBinaryTree(root->left);
123         if(root->right != NULL)
124             deleteBinaryTree(root->right);
125         delete root;
126         return;
127     }
128
129     void deleteNaryTree(Node* root) {
130         if(root == NULL) return;
131         for(int i=0; i<root->children.size(); i++) {
132             deleteNaryTree(root->children[i]);
133         }
134         delete root->children[i];
135     }

```

```

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     return 0;
146 }
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 32
34 33
35 34
36 35
37 36
38 37
39 38
40 39
41 40
42 */

```

## 12 reference

### 12.1 hw6

```

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]=integer[i]+other.integer[i];
22             if(result.integer[i]>=10){
23                 result.integer[i]-=10;
24                 result.integer[i+1]++;
25             }
26         }
27         return result;
28     }
29
30     HugeInt operator-(const HugeInt& other)const{
31         HugeInt result("");
32         for(int i=0;i<40;i++){
33             result.integer[i]=integer[i]-other.integer[i];
34             if(result.integer[i]<0){
35                 result.integer[i]+=10;
36                 result.integer[i+1]--;
37             }
38         }
39         return result;
40     }
41
42     HugeInt operator*(const HugeInt& other)const{
43         HugeInt result("");
44         for(int i=0;i<40;i++){
45             for(int j=0;j<40;j++){
46                 result.integer[i+j]+=integer[i]*other.integer[j];
47             }
48             for(int i=0;i<40;i++){
49                 result.integer[i+1]+=result.integer[i]/10;
50                 result.integer[i]%=10;
51             }
52             return result;
53         }
54
55     HugeInt operator/(const HugeInt& other)const{
56         HugeInt result("");
57         HugeInt remainder("0");
58         HugeInt num("10");
59         for(int i=39;i>=0;i--){
60             if(i==39) remainder.integer[0]=integer[i];
61             else{
62                 remainder=remainder*num;
63                 remainder.integer[0]=integer[i];
64             }
65             int quotient=0;
66             while(remainder>other){
67                 remainder=remainder-other;
68                 quotient++;
69             }
70             if(remainder==other){
71                 remainder=remainder-other;
72                 quotient++;
73
74                 result.integer[i]=quotient;
75             }
76             return result;
77         }
78
79     HugeInt operator%(const HugeInt& other)const{
80         HugeInt remainder("0");
81         HugeInt value("0");
82         HugeInt num("10");
83         for(int i=39;i>=0;i--){
84             remainder=remainder*num;
85             remainder.integer[0]=integer[i];
86             while(remainder>other){
87                 remainder=remainder-other;
88                 if(remainder==other) remainder=value;
89             }
90             return remainder;
91         }
92
93     bool operator>(const HugeInt& other)const{
94         for(int i=39;i>=0;i--){
95             if(integer[i]>other.integer[i]) return true;
96             else if(integer[i]<other.integer[i]) return false;
97         }
98         return false;
99     }
100
101     bool operator<(const HugeInt& other)const{
102         for(int i=39;i>=0;i--){
103             if(integer[i]<other.integer[i]) return true;
104             else if(integer[i]>other.integer[i]) return false;
105         }
106         return false;
107     }
108
109     bool operator==(const HugeInt& other)const{
110         for(int i=39;i>=0;i--){
111             if(integer[i]!=other.integer[i]) return false;
112         }
113         return true;
114     }
115
116     bool operator>=(const HugeInt& other)const{
117         for(int i=39;i>=0;i--){
118             if(integer[i]<other.integer[i]) return false;
119             return true;
120         }
121
122     bool operator<=(const HugeInt& other)const{
123         for(int i=39;i>=0;i--){
124             if(integer[i]>other.integer[i]) return false;
125             return true;
126         }
127
128     bool operator!=(const HugeInt& other)const{
129         return !(*this==other);
130     }
131
132     friend istream& operator>>(istream& in, HugeInt& hugeInt){
133         string s;
134         in>>s;
135         hugeInt=HugeInt(s);
136         return in;
137     }
138
139     friend ostream& operator<<(ostream& out, const HugeInt& hugeInt){
140         bool isLeadingZero=true;
141         for(int i=39;i>=0;i--){

```

```
139         if(hugeInt.integer[i])
140             isLeadingZero=false;
141         if(!isLeadingZero)
142             out<<hugeInt.integer[i];
143     }
144     if(isLeadingZero) out<<0;
145     return out;
146 }
147
148 void print(HugeInt a,HugeInt b){
149     if(a>b) cout<<a<<" > "<<b<<endl;
150     else if(a<b) cout<<a<<" < "<<b<<endl;
151     else cout<<a<<" = "<<b<<endl;
152     cout<<a<<" + "<<b<<" = "<<a+b<<endl;
153     if(a>b) cout<<a<<" - "<<b<<" = "<<a-b<<endl;
154     else if(a<b) cout<<a<<" - "<<b<<" =
155         -"<<b-a<<endl;
156     else cout<<a<<" - "<<b<<" = "<<a-b<<endl;
157     cout<<a<<" * "<<b<<" = "<<a*b<<endl;
158     cout<<a<<" / "<<b<<" = "<<a/b<<endl;
159     cout<<a<<" % "<<b<<" = "<<a%b<<endl;
160 }
161
162 int main(){
163     string x,y;
164     int cnt=0;
165     while(cin>>x>>y){
166         HugeInt x1(x);
167         if(cnt++) cout<<endl;
168         x1.print(x,y);
169     }
170 }
```