2 2

4

5 5

6 6

12

12

13

13

13

13

14 14

15

15

16

Contents

1	ubuntu 1.1 run
2	Basic 2.1 ascii 2.2 limits
3	字串 3.1 最長迴文子字串
4	STL 4.1 priority_queue
5	sort 5.1 大數排序 5.2 bubble sort
6	math 6.1 質數與因數
7	algorithm 7.1 basic
8	動態規劃 8.1 LCS 和 LIS
9	graph 9.1 graph
10	Section2 10.1 thm
11	space 11.1 s
12	reference 12.1 assert.cpp 12.2 find.cpp 12.3 hw6.cpp 12.4 print.cpp 12.5 rectangle.cpp 12.6 rvec.cpp 12.7 sort.cpp 12.8 RationalNumber.cpp 12.9 SortFunctionTemplate.cpp 12.10minmax.cpp

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
            echo Running
            ./$1 < $1.in > $1.out
            echo END
```

Basic

2.1 ascii

int	char	int	char	int	char
32		64	@	96	•
33	!	65	Α	97	а
34	"	66	В	98	b
35	#	67	С	99	С
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	,	71	G	103	g
40	(72	Н	104	h
41)	73	I	105	i
42	*	74	J	106	j
43	+	<i>75</i>	K	107	k
44	,	76	L	108	1
45	-	77	М	109	m
46		78	N	110	n
47	/	79	0	111	0
48	0	80	P	112	р
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	S
<i>52</i>	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	1	90	Z	122	Z
59	;	91	Γ	123	{
60	<	92	\	124	1
61	=	93]	125	}
62	>	94	٨	126	~
63	?	95	_		
	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 55 56 57 58 59 60 61 62	32 33 ! 34 " 35 # 36 \$ 37 % 38 & 39 ' 40 (41) 42 * 44 , 45 - 46 . 47 / 48 0 49 1 50 2 51 3 52 4 53 5 54 6 55 7 56 8 57 9 58 : 59 ; 60 < 61 = 62 >	32 64 33 ! 65 34 " 66 35 # 67 36 \$ 68 37 % 69 38 & 70 39 ' 71 40 (72 41)) 73 42 * 74 43 + 75 44 , 76 45 - 77 46 . 78 47 / 79 48 0 80 49 1 81 50 2 82 51 3 83 52 4 84 53 5 85 54 6 86 55 7 87 56 8 88 57 9 89 58 : 90 59 ; 91 60 < 92 61 = 93 62 > 94	32	32 64 @ 96 33 ! 65 A 97 34 " 66 B 98 35 # 67 C 99 36 \$ 68 D 100 37 % 69 E 101 38 & 70 F 102 39 ' 71 G 103 40 (72 H 104 41) 73 I 105 42 * 74 J 106 43 + 75 K 107 44 , 76 L 108 45 - 77 M 109 46 . 78 N 110 47 / 79 0 111 48 0 80 P 112 49 1 81 Q 113 50 2 82 R 114

2.2 limits

```
1 [Type]
                       [size]
                                    [range]
   2 char
                                  127 to -128
                         1
                                  127 to -128
   3 signed char
     unsigned char
                                  0 to 255
                         1
     short
                         2
                                   32767 to -32768
     int
                                   2147483647 to -2147483648
     unsigned int
                                   0 to 4294967295
15 8 long
                                   2147483647 to -2147483648
  9 unsigned long
                         4
                                   0 to 18446744073709551615
  10 long long
                         8
16 11
                9223372036854775807 to -9223372036854775808
  12 double
                              1.79769e+308 to 2.22507e-308
                         8
  13 long double
                         16
                              1.18973e+4932 to 3.3621e-4932
  14 float
                         4
                                 3.40282e+38 to 1.17549e-38
  15 unsigned long long
                         8
                                   0 to 18446744073709551615
                         32
  16 string
```

字串

3.1 最長迴文子字串

```
1| #include <bits/stdc++.h>
 #define T(x) ((x) % 2 ? s[(x) / 2] : '.')
3
 using namespace std;
5
 string s;
6 int n;
8
 int ex(int 1, int r) {
  int i = 0;
```

```
while(l - i \ge 0 \&\& r + i < n \&\& T(l - i) == T(r + i)
10
         i)) i++;
11
     return i;
12 }
13
14 int main() {
    cin >> s;
15
     n = 2 * s.size() + 1;
17
18
     int mx = 0;
19
    int center = 0;
     vector<int> r(n);
20
21
     int ans = 1;
     r[0] = 1;
22
23
     for(int i = 1; i < n; i++) {</pre>
       int ii = center - (i - center);
24
       int len = mx - i + 1;
25
26
       if(i > mx) {
         r[i] = ex(i, i);
27
28
         center = i;
         mx = i + r[i] - 1;
29
30
       } else if(r[ii] == len) {
         r[i] = len + ex(i - len, i + len);
31
         center = i;
32
         mx = i + r[i] - 1;
33
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;</pre>
```

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:
     priority_queue <int> pq;
  把元素 x 加進 priority_queue:
     pq.push(x);
5
  讀取優先權最高的值:
6
7
     x = pq.top();
                            //讀取後刪除
     pq.pop();
  判斷是否為空的priority_queue:
9
                            //回傳 true
10
     pq.empty()
11
     pq.size()
12 | 如需改變priority_queue的優先權定義:
13
     priority_queue <T> pq;
                            //預設由大到小
     priority_queue<T, vector<T>, greater<T> > pq;
14
                            //改成由小到大
15
     priority_queue <T, vector <T>, cmp> pq; //cmp
16
```

4.2 queue

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

4.3 deque

```
1 deque 是 C++ 標準模板函式庫
2
     (Standard Template Library, STL)
     中的雙向佇列容器(Double-ended Queue),
3
     跟 vector 相似,不過在 vector
4
        中若是要添加新元素至開端,
     其時間複雜度為 O(N),但在 deque 中則是 O(1)。
5
     同樣也能在我們需要儲存更多元素的時候自動擴展空間,
6
     讓我們不必煩惱佇列長度的問題。
7
8 dq.push_back() //在 deque 的最尾端新增元素
 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)
    position: 插入元素的 index 值
16
    n: 元素插入次數
17
    val: 插入的元素值
18
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。
              //清空整個 deque 佇列。
20 dg.clear()
21 dq.size()
              //檢查 deque 的尺寸
              //如果 deque 佇列為空返回 1;
22 dq.empty()
                若是存在任何元素,則返回0
23
24 dq.begin()
              //返回一個指向 deque 開頭的迭代器
25 dq.end()
              //指向 deque 結尾,
26
               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

27

```
1 map:存放 key-value pairs 的映射資料結構,
       會按 key 由小到大排序。
2
3 元素存取
4 operator[]:存取指定的[i]元素的資料
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
28 count():回傳指定元素出現的次數
29 find(): 查找一個元素
31 //實作範例
32 | #include <bits/stdc++.h>
33 using namespace std;
35 int main(){
36
37
      //declaration container and iterator
38
      map<string, string> mp;
39
      map<string, string>::iterator iter;
      map<string, string>::reverse_iterator iter_r;
40
41
      //insert element
42
43
      mp.insert(pair<string, string>("r000",
          "student_zero"));
44
      mp["r123"] = "student_first";
45
      mp["r456"] = "student_second";
46
47
48
      //traversal
      for(iter = mp.begin(); iter != mp.end(); iter++)
49
          cout << iter -> first << " "<< iter -> second << endl;</pre>
50
      for(iter_r = mp.rbegin(); iter_r != mp.rend();
51
          iter_r++)
          cout<<iter_r->first<<"
52
              "<<iter_r->second<<endl;
53
      //find and erase the element
54
55
      iter = mp.find("r123");
      mp.erase(iter);
56
57
      iter = mp.find("r123");
58
59
60
      if(iter != mp.end())
        cout << "Find, the value is
61
             "<<iter->second<<endl;
62
      else
         cout << "Do not Find" << endl;</pre>
63
64
      return 0;
65
```

```
66 }
67
68 //map統計數字
  #include < bits / stdc++.h>
70
  using namespace std;
72
  int main(){
    ios::sync_with_stdio(0),cin.tie(0);
73
    long long n,x;
74
75
     cin>>n;
     map <int,int> mp;
76
     while(n--){
77
78
       cin>>x;
79
       ++mp[x];
    }
80
     for(auto i:mp) cout<<i.first<< " "<<i.second<<endl;</pre>
81
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
  宣告:
3
     set <int> st;
4
5
  把元素 x 加進 set:
6
7
     st.insert(x);
8
  檢查元素 x 是否存在 set 中:
9
10
      st.count(x):
11
  刪除元素 x:
12
     st.erase(x); // 可傳入值或iterator
13
14
  清空集合中的所有元素:
15
     st.clear():
16
17
  取值: 使用iterator
18
19
     x = *st.begin();
             // set中的第一個元素(最小的元素)。
20
21
      x = *st.rbegin();
22
             // set中的最後一個元素(最大的元素)。
23
  判斷是否為空的set:
24
      st.empty() 回傳true
25
      st.size() 回傳零
26
27
  常用來搭配的member function:
28
29
      st.count(x);
      auto it = st.find(x);
30
31
         // binary search, O(log(N))
      auto it = st.lower_bound(x);
32
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:
```

```
st.erase(val); 會刪除所有值為 val 的元素。
6
    st.erase(st.find(val)); 只刪除第一個值為 val
        的元素。
```

unordered set

```
1 unordered_set 的實作方式通常是用雜湊表(hash table),
3 相對的代價是消耗較多的記憶體,空間複雜度較高,
  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
  int main() {
22
23
      unordered_set < int > myunordered_set = {3, 1};
      myunordered_set.insert(2);
24
25
      myunordered_set.insert(5);
26
      myunordered_set.insert(4);
      myunordered_set.insert(5);
27
28
      myunordered_set.insert(4);
29
30
      for (const auto &s : myunordered_set) {
          cout << s << " ";
31
32
      }
      cout << "\n";
33
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);
      for (const auto &s : myunordered_set) {
51
          cout << s << " ";
52
53
      cout << "\n";
54
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);
  myunordered_set.insert(4);
72 myunordered_set.insert(6);
  cout << myunordered_set.count(4) << "\n"; // 1</pre>
73
  cout << myunordered_set.count(8) << "\n"; // 0
75
76
  判斷 unordered_set 容器是否為空
77
  #include <iostream>
  #include <unordered set>
78
79
  int main() {
80
       unordered_set < int > myunordered_set;
81
82
       myunordered_set.clear();
83
84
       if (myunordered_set.empty()) {
           cout << "empty\n";</pre>
85
86
       } else {
           cout << "not empty, size is "<<</pre>
87
                myunordered_set.size() << "\n";</pre>
88
       }
89
90
       return 0;
91 }
```

單調隊列 4.9

46

```
1 //單調隊列
  "如果一個選手比你小還比你強,你就可以退役了。"--單調隊列
  example 1
  給出一個長度為 n 的數組,
  輸出每 k 個連續的數中的最大值和最小值。
9 //寫法1
10
  #include <bits/stdc++.h>
  #define maxn 1000100
12 using namespace std:
13 int q[maxn], a[maxn];
14 int n, k;
15
16
  void getmin() {
      // 得到這個隊列裡的最小值,直接找到最後的就行了
17
      int head = 0, tail = 0;
18
19
      for (int i = 1; i < k; i++) {
          while (head <= tail && a[q[tail]] >= a[i])
20
              tail--
21
          q[++tail] = i;
22
      }
23
      for (int i = k; i <= n; i++) {</pre>
          while (head <= tail && a[q[tail]] >= a[i])
24
              tail--
          q[++tail] = i;
25
          while (q[head] <= i - k) head++;</pre>
26
          cout << a[q[head]] << " ";
27
28
      }
29
  }
30
31
  void getmax() { // 和上面同理
32
      int head = 0, tail = 0;
33
      for (int i = 1; i < k; i++) {
          while(head<=tail&&a[q[tail]]<=a[i])tail--;</pre>
34
35
          q[++tail] = i;
36
      for (int i = k; i <= n; i++) {</pre>
37
38
          while(head<=tail&&a[q[tail]]<=a[i])tail--;</pre>
39
          q[++tail] = i;
          while (q[head] <= i - k) head++;</pre>
40
          cout << a[q[head]] << " ";
41
42
      }
43
  }
45 int main() {
      cin>>n>>k; //每k個連續的數
```

```
for (int i = 1; i <= n; i++) cin>>a[i];
47
48
        getmin():
49
        cout << '\n';
        getmax();
50
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) {
            for(int i=0; i<n; i++) cin >> a[i];
66
            deque<int> dq;
67
            for(int i=0; i<n; i++){</pre>
68
                 while(dq.size() && dq.front()<=i-k)</pre>
69
70
                     dq.pop_front();
71
                 while(dq.size() && a[dq.back()]>a[i])
72
                     dq.pop_back();
73
                 dq.push_back(i);
                 if(i==k-1) cout<<a[dq.front()];</pre>
74
                if(i>k-1) cout<< ' '<<a[dq.front()];</pre>
75
76
77
            if(k>n) cout << a[dq.front()];</pre>
            cout << '\n';
78
79
            while(dq.size()) dq.pop_back();
            for(int i=0; i<n; i++){</pre>
80
81
                 while(dq.size() && dq.front()<=i-k)</pre>
82
                     dq.pop_front();
83
                 while(dq.size() && a[dq.back()]<a[i])</pre>
                     dq.pop_back();
84
85
                 dq.push_back(i);
                 if(i==k-1) cout<<a[dq.front()];</pre>
86
                 if(i>k-1) cout<< ' '<<a[dq.front()];</pre>
87
88
89
            if(k>n) cout << a[dq.front()];</pre>
            cout << '\n';
90
91
92
        return 0;
93 }
94
95
96
   example 2
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。
   若前面的數不足 m 項則從第 1
99
        個數開始,若前面沒有數則輸出 0
100
101 #include <bits/stdc++.h>
102 using namespace std;
103 #define re register int
104 #define INF 0x3f3f3f3f
105 #define ll long long
106 #define maxn 2000009
   #define maxm
108
   inline 11 read() {
       11 x=0, f=1;
109
        char ch=getchar();
110
        while(ch<'0'||ch>'9'){
111
        if(ch=='-') f=-1;
112
113
        ch=getchar();
114
        while(ch>= '0'&&ch<='9'){</pre>
115
        x=(x<<1)+(x<<3)+(11)(ch-'0');
116
117
        ch=getchar();
118
119
        return x*f;
120 }
121 int n,m,k,tot,head,tail;
```

```
122 int a[maxn],q[maxn];
   int main() {
123
       n=read(), m=read();
124
       for(int i=1;i<=n;i++) a[i]=read();</pre>
125
       head=1,tail=0;//起始位置為1
126
            因為插入是q[++tail]所以要初始化為0
127
       for(int i=1;i<=n;i++)</pre>
         //每次隊首的元素就是當前的答案
128
129
130
            cout << a[q[head]] << endl;</pre>
            while(i-q[head]+1>m&&head<=tail)//維護隊首
131
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
  while True:
3
4
    trv:
                              # 有幾筆數字需要排序
5
     n = int(input())
                              # 建立空串列
6
     arr = []
7
      for i in range(n):
       arr.append(int(input())) # 依序將數字存入串列
8
                              # 串列排序
9
      arr.sort()
10
      for i in arr:
                           # 依序印出串列中每個項目
11
       print(i)
12
    except:
     break
13
```

5.2 bubble sort

```
1 #include <bits/stdc++.h>
  using namespace std;
4
  int main() {
    int n:
     cin>>n;
     int a[n], tmp;
7
8
     for(int i=0; i<n; i++) cin>>a[i];
9
     for(int i=n-1; i>0; i--) {
       for(int j=0; j<=i-1; j++) {</pre>
10
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]<<" ";</pre>
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;
  for(int i=2;i*i<=n;i++){</pre>
      if(isprime[i])
8
9
          for(int j=i*i;j<=n;j+=i) isprime[j]=0;</pre>
10 }
11
12
13
  因數
14
15
16
  最大公因數 O(log(min(a,b)))
17 int GCD(int a, int b)
18 {
19
      if (b == 0) return a;
20
      return GCD(b, a % b);
21 | }
22
23 質因數分解
24
25 void primeFactorization(int n)
26 {
27
      for (int i = 0; i < (int)p.size(); ++i)</pre>
28
           if (p[i] * p[i] > n)
29
30
               break;
31
          if (n % p[i])
32
               continue;
           cout << p[i] << ' ';
33
           while (n % p[i] == 0)
34
35
              n /= p[i];
36
      if (n != 1)
37
          cout << n << ' ';
38
39
       cout << '\n';
40 | }
41
42|歌德巴赫猜想
43 | solution : 把偶數 N (6≤N≤10^6) 寫成兩個質數的和。
44 #include <iostream>
45 #include <cstdio>
46 using namespace std;
47 #define N 20000000
48 int ox[N], p[N], pr;
49
  void PrimeTable(){
50
51
      ox[0] = ox[1] = 1;
      pr = 0;
52
53
       for (int i = 2; i < N; i++){
          if (!ox[i]) p[pr++] = i;
54
           for (int j = 0;i*p[j]<N&&j < pr; j++)</pre>
55
56
               ox[i*p[j]] = 1;
57
      }
58 }
59
  int main(){
      PrimeTable():
61
62
      int n:
63
          while (cin>>n,n){
               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("02")
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
        cin.tie(nullptr);
81
        cout.tie(nullptr);
82
        ios_base::sync_with_stdio(false);
83
84
85
   bool isPrime(int n){
        FOR(i, 2, n){
86
87
            if (i * i > n)
88
                 return true;
             if (n % i == 0)
89
90
                 return false;
91
92
        return true;
93 }
94
   int main(){
95
        IOS;
96
97
        int n;
98
        cin >> n;
99
        if(isPrime(n)) cout << "1\n";</pre>
        else if(n%2==0||isPrime(n-2)) cout << "2\n";</pre>
100
101
        else cout << "3\n";</pre>
102 }
```

6.2 prime factorization

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

6.3 快速冪

```
1 計算a^b
  #include <iostream>
  #define ll long long
  using namespace std;
6
  const 11 MOD = 1000000007;
  11 fp(11 a, 11 b) {
7
     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;
     cin>>a>>b:
19
20
     cout << fp(a,b);</pre>
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;
6 int phi(){
7
       ans=n:
8
       for(int i=2;i*i<=n;i++)</pre>
9
           if(n%i==0){
10
                ans=ans-ans/i;
11
                while(n%i==0) n/=i;
12
13
       if(n>1) ans=ans-ans/n;
14
       return ans;
15 }
16
17 int main(){
18
     while(cin>>n)
19
         cout << phi() << endl;</pre>
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
16 next_permutation(first, last)
17 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 。
18
19 prev_permutation(first, last)
```

7.2 binarysearch

```
1 #include <bits/stdc++.h>
2 using namespace std;
  int binary_search(vector<int> &nums, int target) {
       int left=0, right=nums.size()-1;
5
6
       while(left<=right){</pre>
7
           int mid=(left+right)/2;
           if (nums[mid]>target) right=mid-1;
9
           else if(nums[mid]<target) left=mid+1;</pre>
10
           else return mid+1;
11
12
       return 0;
13 }
14
15 int main() {
16
    int n, k, x;
    cin >> n >> k;
17
18
     int a[n];
19
     vector<int> v;
20
     for(int i=0 ; i<n ; i++){</pre>
21
       cin >> x;
       v.push_back(x);
22
```

```
23
    for(int i=0 ; i<k ; i++) cin >> a[i];
24
25
    for(int i=0 ; i<k ; i++){</pre>
      cout << binary_search(v, a[i]) << endl;</pre>
26
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
  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
43
44
  /*
45 output
46
  2
47
  5
48
  4
50
  0
  */
```

7.3 prefix sum

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

7.4 差分

```
1 // 差分
2|用途:在區間 [1, r] 加上一個數字v。
3|b[1] += v; (b[0~1] 加上v)
4 b[r+1] -= v; (b[r+1~n] 減去v (b[r] 仍保留v) )
  給的 a[] 是前綴和數列,建構 b[],
  因為 a[i] = b[0] + b[1] + b[2] + ··· + b[i],
  所以 b[i] = a[i] - a[i-1]。
  在 b[1] 加上 v,b[r+1] 減去 v,
  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];
  // a: 前綴和數列, b: 差分數列
15
16 int main(){
     int n, 1, r, v;
```

```
18
      cin >> n;
                                                      56
                                                                   cin >> x;
      for(int i=1; i<=n; i++){</pre>
                                                      57
                                                                   q.push(x);
19
20
         cin >> a[i];
                                                      58
         b[i] = a[i] - a[i-1]; //建構差分數列
21
                                                      59
                                                               long long ans = 0;
                                                      60
                                                                while (q.size() > 1){
22
                                                      61
                                                                   x = q.top();
23
      cin >> 1 >> r >> v;
                                                                   q.pop();
     b[1] += v;
                                                      62
24
     b[r+1] -= v;
                                                      63
                                                                   x += q.top();
25
                                                      64
                                                                   q.pop();
26
      for(int i=1; i<=n; i++){</pre>
                                                      65
                                                                   q.push(x);
27
         b[i] += b[i-1];
                                                      66
                                                                   ans += x;
28
         cout << b[i] << ' ';
                                                      67
29
                                                      68
                                                                cout << ans << endl;</pre>
30
                                                            }
                                                      69
31 }
                                                      70
                                                        }
                                                      71
                                                      72 Commando War
  7.5 greedy
                                                        //problem
                                                      74 有 n 個部下,每個部下要花 Bi 分鐘交待任務,
                                                        再花 Ji 分鐘執行任務,一次只能對一位部下交代任務,
1 // 貪心
                                                        但可以多人同時執行任務,問最少要花多少時間完成任務。
77
3 採取在目前狀態下最好或最佳(即最有利)的選擇。
                                                        //solution
                                                      78
4 貪心演算法雖然能獲得當前最佳解,
                                                        執行時間長的人先交代任務
                                                      79
5 但不保證能獲得最後(全域)最佳解,
                                                      80
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,
  確認無誤再實作。
                                                        #include <bits/stdc++.h>
                                                      82
8
                                                      83
                                                        using namespace std;
9 Scarecrow
                                                        struct Data{
                                                      84
10 //problem
                                                            int b, j;
                                                      85
11 有一個 N×1 的稻田,有些稻田現在有種植作物,
                                                            bool operator <(const Data &rhs) const {</pre>
12 | 為了避免被動物破壞,需要放置稻草人,
                                                      87
                                                                return j > rhs.j;
                                                      88
13 | 稻草人可以保護該塊稻田和左右兩塊稻田,
                                                      89
                                                        };
14 請問最少需要多少稻草人才能保護所有稻田?
                                                      90
15
                                                        int main(){
16 //solutoin
                                                      92
                                                            int n, ti = 0;
  從左到右掃描稻田,如果第 i 塊稻田有作物,
17
                                                            Data a[1005];
                                                      93
  就把稻草人放到第 i+1 塊稻田,
                                                      94
                                                            while (cin >> n, n){
  這樣能保護第 i,i+1,i+2 塊稻田,
19
                                                                for (int i = 0; i < n; ++i)
                                                      95
20 接著從第 i+3 塊稻田繼續掃描。
                                                      96
                                                                   cin >> a[i].b >> a[i].j;
21
                                                      97
                                                                sort(a, a + n);
22 //code
                                                      98
                                                                int ans = 0, sum = 0;
23 #include <bits/stdc++.h>
                                                                for (int i = 0; i < n; ++i){
                                                      99
24 using namespace std;
                                                     100
                                                                   sum += a[i].b;
25 int main(){
                                                     101
                                                                   ans = max(ans, sum + a[i].j);
26
      string s;
                                                     102
27
      int i, n, t, tc = 1;
                                                                cout << "Case "<<++ti<<": "<<ans<<'\n';
                                                     103
28
      cin >> t;
                                                            }
                                                     104
      while (t--){
29
                                                     105 }
30
         cin >> n >> s;
                                                     106
31
         int nc = 0;
                                                     107 刪數字問題
         for (i = 0; i < n; i++)
32
                                                        //problem
                                                     108
33
             if (s[i] == '.')i += 2, nc++;
                                                        給定一個數字 N(≤10^100),需要刪除 K 個數字,
                                                     109
         cout << "Case "<<tc++<<": "<<nc<<endl;
34
                                                     110
                                                        請問刪除 K 個數字後最小的數字為何?
35
                                                     111
36 }
                                                        //solution
                                                     112
37
                                                     113 刪除滿足第 i 位數大於第 i+1 位數的最左邊第 i 位數,
38 霍夫曼樹的變形題
                                                     114 扣除高位數的影響較扣除低位數的大。
                                                     115
40 給定 N 個數,每次將兩個數 a,b 合併成 a+b,
                                                     116
                                                        //code
41 | 只到最後只剩一個數,合併成本為兩數和,
                                                     117
                                                        int main()
42 問最小合併成本為多少。
                                                     118
                                                        {
43
                                                     119
                                                            string s;
  //solution
44
                                                     120
                                                            int k;
45 每次將最小的兩數合併起來。
                                                     121
                                                            cin >> s >> k;
46
                                                     122
                                                            for (int i = 0; i < k; ++i){
                                                     123
                                                                if ((int)s.size() == 0) break;
48 #include <bits/stdc++.h>
                                                                int pos = (int)s.size() - 1;
                                                     124
49 using namespace std;
                                                                for (int j = 0; j < (int)s.size() - 1; ++j){</pre>
                                                     125
50 int main()
                                                                   if (s[j] > s[j + 1]){
                                                     126
51 \ {
                                                     127
                                                                       pos = j;
52
                                                     128
                                                                       break;
53
      while (cin >> n, n){
                                                                   }
                                                     129
```

131

s.erase(pos, 1);

54

55

while (n--){

priority_queue < int , vector < int > , greater < int >>

```
132
                                                       207
                                                              for (int i=1;i<=n;++i) cin>>a[i];
      while ((int)s.size() > 0 && s[0] == '0')
                                                              int i = 1, ans = 0;
                                                       208
133
          s.erase(0, 1);
                                                              while (i <= n){</pre>
134
                                                       209
      if ((int)s.size()) cout << s << '\n';</pre>
135
                                                       210
                                                                  int R=min(i+r-1, n), L=max(i-r+1, 0)
136
      else cout << 0 << '\n';
                                                       211
                                                                  int nextR=-1;
                                                                  for (int j = R; j >= L; --j){}
137
                                                       212
                                                                     if (a[j]){
138
                                                       213
139
                                                       214
                                                                         nextR = j;
140 區間覆蓋長度
                                                       215
                                                                         break;
                                                                     }
                                                       216
141 //problem
142 給定 n 條線段區間為 [Li,Ri],
                                                       217
                                                                  }
                                                                  if (nextR == -1){
  請問這些線段的覆蓋所覆蓋的長度?
                                                       218
143
                                                                     ans = -1;
                                                       219
144
                                                                     break:
                                                       220
145
   //solution
                                                       221
                                                                  }
146 先將所有區間依照左界由小到大排序,
                                                       222
                                                                  ++ans;
147 左界相同依照右界由小到大排序,
                                                       223
                                                                  i = nextR + r;
148 用一個變數 R 紀錄目前最大可以覆蓋到的右界。
                                                       224
                                                              }
149 如果目前區間左界 ≤R,代表該區間可以和前面的線段合併。
                                                       225
                                                              cout << ans << '\n';
150
                                                       226
                                                          }
151 //code
                                                       227
152 struct Line
                                                       228
153
                                                       229 最多不重疊區間
      int L, R;
154
                                                       230 //problem
155
      bool operator < (const Line &rhs) const
                                                       231 | 給你 n 條線段區間為 [Li,Ri],
156
                                                          請問最多可以選擇幾條不重疊的線段(頭尾可相連)?
                                                       232
          if (L != rhs.L) return L < rhs.L;</pre>
157
                                                       233
158
          return R < rhs.R;</pre>
                                                       234
                                                          //solution
      }
159
                                                       235 依照右界由小到大排序,
160
  };
                                                          每次取到一個不重疊的線段,答案 +1。
                                                       236
161
                                                       237
  int main(){
162
                                                       238
                                                          //code
163
      int n;
                                                       239
                                                          struct Line
      Line a[10005];
164
                                                       240
165
      while (cin >> n){
                                                       241
                                                              int L, R;
          for (int i = 0; i < n; i++)
166
                                                              bool operator<(const Line &rhs) const {</pre>
                                                       242
167
              cin >> a[i].L >> a[i].R;
                                                                  return R < rhs.R;</pre>
                                                       243
168
          sort(a, a + n);
          int ans = 0, L = a[0].L, R = a[0].R;
                                                       244
169
                                                       245
                                                          };
          for (int i = 1; i < n; i++){</pre>
170
              if (a[i].L < R) R = max(R, a[i].R);</pre>
                                                       246
171
                                                          int main(){
                                                       247
172
              else{
                                                       248
                                                              int t;
                  ans += R - L;
173
                                                              cin >> t;
                                                       249
                  L = a[i].L;
174
                                                       250
                                                              Line a[30];
175
                  R = a[i].R;
                                                              while (t--){
                                                       251
176
                                                                  int n = 0;
                                                       252
177
                                                       253
                                                                  while (cin>>a[n].L>>a[n].R, a[n].L||a[n].R)
          cout << ans + (R - L) << ' \ n';
178
                                                       254
179
                                                                     ++n;
                                                       255
                                                                  sort(a, a + n);
180
  }
                                                                  int ans = 1, R = a[0].R;
                                                       256
181
                                                                  for (int i = 1; i < n; i++){
                                                       257
182
                                                       258
                                                                     if (a[i].L >= R){
   最小區間覆蓋長度
183
                                                       259
                                                                         ++ans:
184 //problem
                                                       260
                                                                         R = a[i].R;
185 | 給定 n 條線段區間為 [Li,Ri],
                                                                     }
                                                       261
186 請問最少要選幾個區間才能完全覆蓋 [0,S]?
                                                       262
                                                                  cout << ans << ' \setminus n';
                                                       263
188 //solution
                                                       264
                                                              }
265
                                                          }
190 對於當前區間 [Li,Ri],要從左界 >Ri 的所有區間中,
                                                       266
191 找到有著最大的右界的區間,連接當前區間。
192
                                                       268 區間選點問題
193
   //problem
                                                          //problem
                                                       269
   長度 n 的直線中有數個加熱器,
194
                                                       270 給你 n 條線段區間為 [Li,Ri],
   在 x 的加熱器可以讓 [x-r,x+r] 內的物品加熱,
                                                          請問至少要取幾個點才能讓每個區間至少包含一個點?
                                                       271
   問最少要幾個加熱器可以把 [0,n] 的範圍加熱。
196
                                                       272
197
                                                       273
                                                          //solution
198 //solution
                                                       274 將區間依照右界由小到大排序, R=第一個區間的右界,
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,
                                                          遍歷所有區段,如果當前區間左界>R ,
  更新已加熱範圍,重複上述動作繼續尋找加熱器。
                                                       276 代表必須多選一個點 (ans+=1),並將 R=當前區間右界。
201
                                                       277
   //code
202
                                                       278 //problem
203
  int main(){
                                                       279 給定 N 個座標,要在 x 軸找到最小的點,
204
      int n. r:
                                                       280 讓每個座標至少和一個點距離 ≤ D。
205
      int a[1005];
                                                       281
```

//solution

282

206

cin >> n >> r;

```
359 就從目前選擇的工作中,移除耗時最長的工作。
283 以每個點 (xi,yi) 為圓心半徑為 D 的圓 C,
   求出 C 和 x 軸的交點 Li, Ri, 題目轉變成區間選點問題。
                                                       360
284
285
                                                          上述方法為 Moore-Hodgson s Algorithm。
                                                       361
  //code
                                                       362
287 struct Line
                                                       363
                                                          //problem
288 {
                                                       364 | 給定烏龜的重量和可承受重量,問最多可以疊幾隻烏龜?
289
      int L, R;
                                                       365
      bool operator<(const Line &rhs) const {</pre>
290
                                                          和最少延遲數量問題是相同的問題,只要將題敘做轉換。
                                                       366
          return R < rhs.R;</pre>
291
                                                       367
292
                                                          工作處裡時長→ 烏龜重量
                                                       368
293
  };
                                                          工作期限 → 烏龜可承受重量
                                                       369
294
                                                       370 多少工作不延期 → 可以疊幾隻烏龜
   int main(){
295
                                                       371
296
      int t;
                                                       372
                                                          //code
      cin >> t:
297
                                                       373
                                                          struct Work{
298
      Line a[30];
                                                       374
                                                              int t, d;
      while (t--){
299
                                                       375
                                                              bool operator<(const Work &rhs) const {</pre>
300
          int n = 0;
                                                                 return d < rhs.d;</pre>
                                                       376
301
          while (cin>>a[n].L>>a[n].R, a[n].L||a[n].R)
                                                       377
302
                                                       378
                                                          };
303
          sort(a, a + n);
                                                       379
          int ans = 1, R = a[0].R;
304
                                                          int main(){
                                                       380
          for (int i = 1; i < n; i++){
305
                                                       381
                                                              int n = 0;
              if (a[i].L >= R){
306
                                                       382
                                                              Work a[10000];
307
                  ++ans:
                                                       383
                                                              priority_queue<int> pq;
                  R = a[i].R;
308
                                                       384
                                                              while(cin >> a[n].t >> a[n].d)
              }
309
                                                                 ++n:
                                                       385
          }
310
                                                       386
                                                              sort(a, a + n);
311
          cout << ans << '\n';
                                                              int sumT = 0, ans = n;
                                                       387
312
                                                       388
                                                              for (int i = 0; i < n; ++i){
313 }
                                                       389
                                                                  pq.push(a[i].t);
314
                                                                  sumT += a[i].t:
                                                       390
                                                                  if(a[i].d<sumT){</pre>
316 最小化最大延遲問題
                                                                     int x = pq.top();
                                                       392
317 //problem
                                                       393
                                                                     pq.pop();
318 給定 N 項工作,每項工作的需要處理時長為 Ti,
                                                       394
                                                                     sumT -= x;
319 期限是 Di, 第 i 項工作延遲的時間為 Li=max(0, Fi-Di),
                                                       395
                                                                      --ans:
320 原本Fi 為第 i 項工作的完成時間,
                                                       396
                                                                 }
  |求一種工作排序使 maxLi 最小。
                                                              }
321
                                                       397
                                                              cout << ans << '\n';
322
                                                       398
323
  //solution
                                                       399 }
324 按照到期時間從早到晚處理。
                                                       400
                                                       401 任務調度問題
325
326 //code
                                                          //problem
                                                       402
327 struct Work
                                                       403|給定 N 項工作,每項工作的需要處理時長為 Ti,
328
                                                       404 期限是 Di,如果第 i 項工作延遲需要受到 pi 單位懲罰,
329
      int t, d;
                                                       405
                                                          請問最少會受到多少單位懲罰。
      bool operator<(const Work &rhs) const {</pre>
330
                                                       406
331
          return d < rhs.d;</pre>
                                                          //solution
                                                       407
332
                                                          依照懲罰由大到小排序,
                                                       408
333
  };
                                                          每項工作依序嘗試可不可以放在 Di-Ti+1, Di-Ti,...,1,0,
334
                                                          如果有空閒就放進去,否則延後執行。
                                                       410
335
   int main(){
                                                       411
336
      int n;
                                                          //problem
337
      Work a[10000];
                                                       413 | 給定 N 項工作,每項工作的需要處理時長為 Ti,
338
      cin >> n;
                                                       414 期限是 Di,如果第 i 項工作在期限內完成會獲得 ai
      for (int i = 0; i < n; ++i)
339
          cin >> a[i].t >> a[i].d;
                                                              單位獎勵,
340
341
      sort(a, a + n);
                                                          請問最多會獲得多少單位獎勵。
                                                       415
      int maxL = 0, sumT = 0;
342
                                                       416
      for (int i = 0; i < n; ++i){
343
                                                       417
                                                          //solution
          sumT += a[i].t;
344
                                                          和上題相似,這題變成依照獎勵由大到小排序。
                                                       418
345
          maxL = max(maxL, sumT - a[i].d);
346
      }
                                                          //code
                                                       420
347
      cout << maxL << '\n';
                                                       421
                                                          struct Work
348 }
                                                       422
349
                                                       423
                                                              int d, p;
350
                                                       424
                                                              bool operator<(const Work &rhs) const {</pre>
351 最少延遲數量問題
                                                                 return p > rhs.p;
                                                       425
352 //problem
                                                       426
353 給定 N 個工作,每個工作的需要處理時長為 Ti,
                                                       427 };
354 期限是 Di, 求一種工作排序使得逾期工作數量最小。
                                                       428
                                                       429
                                                          int main(){
355
                                                       430
356
   //solution
                                                              int n:
                                                              Work a[100005];
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34
                                                              bitset<100005> ok;
358 依序放入工作列表中,如果發現有工作預期,
                                                       432
```

19

20

```
433
       while (cin >> n){
           ok.reset();
434
435
           for (int i = 0; i < n; ++i)
               cin >> a[i].d >> a[i].p;
436
437
           sort(a, a + n);
438
           int ans = 0;
           for (int i = 0; i < n; ++i){
439
440
               int j = a[i].d;
                while (j--)
441
                    if (!ok[j]){
442
443
                        ans += a[i].p;
                        ok[i] = true;
444
445
                    }
446
447
           }
           cout << ans << '\n';
448
449
450 }
451
452 多機調度問題
   //problem
   給定 N 項工作,每項工作的需要處理時長為 Ti,
   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
   //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(){
       int n, m;
463
       int a[10000];
464
465
       cin >> n >> m;
       for (int i = 0; i < n; ++i)
466
467
           cin >> a[i];
       sort(a, a + n,greater<int>());
468
       int ans = 0;
469
       priority_queue < int , vector < int > , greater < int >>pq;
470
471
       for (int i = 0; i < m && i < n; ++i){</pre>
472
           ans = max(ans, a[i]);
473
           pq.push(a[i]);
474
475
       for (int i = m; i < n; ++i){
           int x = pq.top();
476
477
           pq.pop();
478
           x += a[i];
479
           ans = max(ans, x);
           pq.push(x);
480
481
       cout << ans << '\n';
482
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 ,
  C 同時為 A,B 的子序列。
3
5 //最長遞增子序列 (LIS)
  給你一個序列 A , 求最長的序列 B ,
6
   B 是一個(非)嚴格遞增序列,且為 A 的子序列。
7
9 //LCS 和 LIS 題目轉換
10 LIS 轉成 LCS
    1. A 為原序列, B=sort(A)
11
    2. 對 A,B 做 LCS
12
13 LCS 轉成 LIS
    1. A, B 為原本的兩序列
14
    2. 最 A 序列作編號轉換,將轉換規則套用在 B
15
16
    3. 對 B 做 LIS
```

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

```
61
        vector<int> nums = {}, result;
                                                                   130
   public:
                                                                                delete root;
                                                                   131
62
        void solve() {
63
                                                                   132
                                                                                return;
                                                                           }
64
                                                                   133
65
            return;
                                                                   134
66
        }
                                                                   135
                                                                           void inorderTraversal(TreeNode* root) {
                                                                                if(root == NULL) return;
67
                                                                   136
68
        TreeNode* buildBinaryTreeUsingDFS(int left, int
                                                                   137
                                                                                inorderTraversal(root->left);
             right) {
                                                                                cout << root -> val << ' ';</pre>
                                                                   138
            if((left > right) || (nums[(left+right)/2] ==
69
                                                                   139
                                                                                inorderTraversal(root->right);
                 null)) return NULL;
                                                                   140
                                                                                return:
            int mid = (left+right)/2;
                                                                           }
70
                                                                   141
71
            TreeNode* current = new TreeNode(
                                                                   142
                                                                      };
                 nums[mid],
72
                                                                   143
73
                 buildBinaryTreeUsingDFS(left,mid-1),
                                                                   144
                                                                      int main() {
                 buildBinaryTreeUsingDFS(mid+1, right));
                                                                   145
74
75
                                                                   146
             return current;
                                                                           return 0;
76
        }
                                                                   147 }
77
78
        TreeNode* buildBinaryTreeUsingBFS() {
79
            int idx = 0:
80
            TreeNode* root = new TreeNode(nums[idx++]);
                                                                              Section2
81
            queue < TreeNode *> q;
            q.push(root);
82
83
            while(idx < nums.size()) {</pre>
                                                                       10.1
                                                                               thm
84
                 if(nums[idx] != null) {
                      TreeNode* left = new
85
                                                                         · 中文測試
                          TreeNode(nums[idx]);
                     q.front()->left = left;
86
                                                                         • \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}
87
                     q.push(left);
                 }
88
89
                 idx++;
90
                 if((idx < nums.size()) && (nums[idx] !=</pre>
                                                                       11
                                                                             space
                      null)) {
91
                      TreeNode* right = new
                          TreeNode(nums[idx]);
                                                                       11.1
                                                                               S
92
                     q.front()->right = right;
                     q.push(right);
93
                                                                     1 /*
                 }
94
                                                                     2
                                                                       1
95
                 idx++:
                                                                       2
                                                                     3
                 q.pop();
96
                                                                       3
97
            }
                                                                       4
98
            return root;
                                                                       5
99
                                                                     7
100
                                                                     8
                                                                       7
101
        Node* buildNAryTree() {
102
            int idx = 2;
                                                                    10
            Node *root = new Node(nums.front());
103
                                                                       10
                                                                    11
            queue < Node *> q;
104
                                                                    12
                                                                       11
            q.push(root);
105
                                                                       12
106
             while(idx < nums.size()) {</pre>
                                                                    13
                                                                    14
                                                                       13
                 while((idx < nums.size()) && (nums[idx]</pre>
107
                                                                    15
                                                                       14
                      != null)) {
                                                                    16
108
                     Node *current = new Node(nums[idx++]);
                     q.front()->children.push_back(current);
                                                                    17
109
                                                                       17
                                                                    18
                     q.push(current);
                                                                    19
                                                                       18
                 }
111
                                                                       19
                                                                    20
112
                 idx++;
                                                                       20
113
                 q.pop();
                                                                    22
                                                                       21
            }
114
                                                                    23
                                                                       22
115
            return root;
                                                                    24
                                                                       23
        }
116
                                                                       24
                                                                    25
117
                                                                    26
        void deleteBinaryTree(TreeNode* root) {
118
                                                                    27
                                                                       26
119
            if(root->left != NULL)
                                                                    28
                 deleteBinaryTree(root->left);
                                                                    29
                                                                       28
            if(root->right != NULL)
120
                                                                       29
                 deleteBinaryTree(root->right);
                                                                    30
                                                                       30
121
             delete root;
                                                                       31
                                                                    32
122
             return;
                                                                    33
                                                                       32
123
        }
                                                                    34
                                                                       33
124
                                                                    35
                                                                       34
125
        void deleteNAryTree(Node* root) {
                                                                    36
                                                                      35
            if(root == NULL) return;
126
                                                                       36
                                                                    37
127
            for(int i=0; i<root->children.size(); i++) {
                                                                       37
                                                                    38
128
                 deleteNAryTree(root->children[i]);
```

39 */

delete root->children[i];

26

27 28

29 30

31

32

33

34

35

36

37 38

39 40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75 76

77 78

79

80

81

82

83 84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

12 reference

12.1 assert.cpp

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

12.2 find.cpp

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

12.3 hw6.cpp

```
1 | #include < iostream >
2
  #include<string>
3 #include < cstring >
4 #include <algorithm>
5 using namespace std;
7
  class HugeInt{
8
  private:
       short integer[40];
10
11
  public:
12
       HugeInt(const string& s){
           memset(integer,0,sizeof(integer));
13
14
            for(int i=0;i<s.length();i++)</pre>
15
                integer[i]=s[s.length()-1-i]-'0';
16
       }
17
       HugeInt operator+(const HugeInt& other)const{
18
19
           HugeInt result("");
           for(int i=0;i<40;i++){</pre>
20
21
                result.integer[i]+=
22
                     integer[i]+other.integer[i];
                if(result.integer[i]>=10){
23
```

```
result.integer[i]-=10;
             result.integer[i+1]++;
        }
    }
    return result;
}
HugeInt operator - (const HugeInt& other)const{
    HugeInt result("");
    for(int i=0;i<40;i++){</pre>
        result.integer[i]+=
             integer[i]-other.integer[i];
        if(result.integer[i]<0){</pre>
             result.integer[i]+=10;
             result.integer[i+1]--;
        }
    }
    return result;
}
HugeInt operator*(const HugeInt& other)const{
    HugeInt result("");
    for(int i=0;i<40;i++)
        for(int j=0;j<40;j++)</pre>
             result.integer[i+j]+=
                 integer[i]*other.integer[j];
    for(int i=0;i<40;i++){</pre>
        result.integer[i+1]+=result.integer[i]/10;
        result.integer[i]%=10;
    return result;
}
HugeInt operator/(const HugeInt& other)const{
    HugeInt result("");
    HugeInt remainder("0");
    HugeInt num("10");
    for(int i=39;i>=0;i--){
        if(i==39) remainder.integer[0]=integer[i];
        else{
             remainder=remainder*num;
             remainder.integer[0]=integer[i];
        int quotient=0;
        while(remainder>other){
             remainder=remainder-other;
             quotient++;
        if(remainder==other){
             remainder=remainder-other;
             quotient++;
        result.integer[i]=quotient;
    return result;
HugeInt operator%(const HugeInt& other)const{
    HugeInt remainder("0");
    HugeInt value("0");
    HugeInt num("10");
    for(int i=39;i>=0;i--){
        remainder=remainder*num;
        remainder.integer[0]=integer[i];
        while(remainder>other)
             remainder=remainder-other;
        if(remainder==other) remainder=value;
    return remainder;
}
bool operator>(const HugeInt& other)const{
    for(int i=39;i>=0;i--){
        if(integer[i]>other.integer[i])
             return true;
        else if(integer[i]<other.integer[i])</pre>
             return false;
```

```
101
102
             return false;
103
        }
104
105
        bool operator < (const HugeInt& other)const{</pre>
             for(int i=39;i>=0;i--){
106
                 if(integer[i]<other.integer[i])</pre>
107
108
                      return true:
                 else if(integer[i]>other.integer[i])
109
110
                      return false;
111
            }
             return false:
112
113
        }
114
115
        bool operator == (const HugeInt& other)const{
             for(int i=39;i>=0;i--)
116
117
                 if(integer[i]!=other.integer[i])
118
                      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])</pre>
125
                      return false;
126
             return true:
127
128
129
        bool operator <= (const HugeInt& other)const{</pre>
130
             for(int i=39; i>=0; i--)
                 if(integer[i]>other.integer[i])
131
132
                      return false;
133
             return true:
134
135
136
        bool operator!=(const HugeInt& other)const{
137
             return !(*this==other);
138
139
140
        friend istream& operator>>(istream& in,
141
                                    HugeInt& hugeInt){
142
             string s;
             in>>s;
143
144
             hugeInt=HugeInt(s);
145
             return in;
146
147
        friend ostream& operator << (ostream& out,</pre>
148
149
                           const HugeInt& hugeInt){
            bool isLeadingZero=true;
150
151
             for(int i=39;i>=0;i--){
152
                 if(hugeInt.integer[i])
153
                      isLeadingZero=false;
154
                 if(!isLeadingZero)
155
                      out << hugeInt.integer[i];</pre>
             if(isLeadingZero) out << 0;</pre>
157
158
             return out;
        }
159
160
        void print(HugeInt a, HugeInt b){
161
             if(a>b) cout<<a<<" > "<<b<<endl;</pre>
162
             else if(a<b) cout<<a<<" < "<<b<<endl;
163
             else cout << a << " = " << b << endl;
164
            cout <<a<<" + "<<b<<" = "<<a+b<<endl;
165
             if(a>b) cout<<a<<" - "<<b<<" = "<<a-b<<endl;</pre>
166
167
             else if(a<b)
                 cout <<a<<" - "<<b<<" = -"<<b-a<<endl;
168
             else cout<<a<<" - "<<b<<" = "<<a-b<<endl;
169
             cout <<a<<" * "<<b<<" = "<<a*b<<endl;
170
             cout <<a<<" / "<<b<<" = "<<a/b<<endl;
171
172
             cout <<a<< " % "<<b<<" = "<<a%b<<endl;
173
174 };
175
176 int main(){
177
        string x,y;
```

```
int cnt=0;
while(cin>>x>>y){
    HugeInt x1(x);
    if(cnt++) cout<<endl;
    x1.print(x,y);
}
</pre>
```

12.4 print.cpp

```
template <typename T>
void printvector(vector<T>& vec){
   for(T &x:vec) cout<<x<<" ";
   cout<<"\n";
   return;
}</pre>
```

12.5 rectangle.cpp

```
1 | #include < iostream >
  #include<iomanip>
  #include < algorithm >
  #include < math.h>
  using namespace std;
  class R {
7
            double area, diagonal;
9
10
       public:
11
            R(){};
            int length, width;
12
13
            R(double 1, double w) {
                 length=1;
14
15
                 width=w;
            }
16
17
            int getA(){
18
                 area=length*width;
19
                 return area;
20
            }
21
            double getD(){
                 diagonal=sqrt(length*length+width*width);
22
                 return diagonal;
23
            }
24
25
  };
26
  bool cmp(R r1, R r2) {
27
       if(r1.getA() == r2.getA())
28
29
            return r1.getD()>r2.getD();
30
       return r1.getA()<r2.getA();</pre>
31 }
  int main() {
33
34
       int n;
35
       cin>>n;
       R* r=new R[n];
36
37
       for(int i=0;i<n;i++) {</pre>
38
            int 1,w;
39
            cin>>l>>w;
40
            r[i]=R(1,w);
41
42
       sort(r,r+n,cmp);
       for(int i=0;i<n;i++){</pre>
43
44
            cout << i +1 << ": ("<< r[i].length << ", "<< r[i].width;
            cout << ") area = "<<r[i].getA();</pre>
45
46
            cout << fixed << setprecision(3)</pre>
47
                 << " diagonal = "<<r[i].getD()<<endl;
48
49
       delete[] r;
50
       return 0;
51 }
```

12.6 rvec.cpp

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

12.7 sort.cpp

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

12.8 RationalNumber.cpp

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

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

```
80
         while(cin>>op){
81
              cin>>g>>r1>>g>>g>>r2>>g;
              if(r1.molecular%r1.denominator) cout<<r1;</pre>
 82
              else cout<<r1.molecular/r1.denominator;</pre>
 83
              cout << " "<<op<< " ";
 84
 85
              if(r2.molecular%r2.denominator) cout<<r2;</pre>
              else cout << r2.molecular/r2.denominator;</pre>
 86
              cout << " = ":
 87
              switch(op){
 88
 89
                   case '+':
 90
                        cout << r1+r2 << endl;</pre>
 91
                        break:
 92
                          1-1:
                        cout << r1 - r2 << end1;
 93
 94
                        break;
                   case '*':
 95
                        cout << r1*r2 << endl;</pre>
 96
 97
                        break;
                   case '/':
98
 99
                        cout << r1/r2 << endl;</pre>
100
                        break:
101
              }
         }
102
103
         return 0;
104 }
```

12.9 SortFunctionTemplate.cpp

```
1 #include < iostream >
2 #include < vector >
  using namespace std;
  template < typename T>
  int p(vector<T>& arr,int low,int high){
6
7
       T pivot=arr[high];
8
       int i=low-1;
       for(int j=low;j<high;j++)</pre>
9
10
            if(arr[j]<pivot){</pre>
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>
   void qsort(vector<T>& arr, int low,int high){
19
20
       if(low<high){</pre>
            int pindex=p(arr,low,high);
21
22
            qsort(arr,low,pindex-1);
            qsort(arr,pindex+1,high);
23
       }
24
25 }
26
27
   template < typename T>
   void mysort(vector<T>& arr){
28
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){
            if(cnt++) cout << " ";
38
39
            cout <<a;
40
41
       cout << endl;
42
       arr.clear();
43 }
44
45
   int main(){
46
       int n,x;
       double d;
47
48
       char c;
```

```
49
        string s;
        while(cin>>n){
50
51
            vector<int> v;
            for(int i=0;i<n;i++){</pre>
52
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++){</pre>
60
                 cin>>d;
61
                 vd.push_back(d);
62
63
            solve(vd);
            cin>>n;
64
65
            vector<char> vc;
66
            for(int i=0;i<n;i++){</pre>
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++){</pre>
                 getline(cin,s);
74
75
                 vs.push_back(s);
            }
76
77
            mysort(vs);
78
            for(auto a:vs) cout<<a<<endl;</pre>
79
            vs.clear():
80
       }
81
        return 0;
82
```

12.10 minmax.cpp

```
#include < bits / stdc ++. h>
2
  using namespace std;
3
4
  int main(){
       vector<int>
5
            num={5,3,3,3,6,6,74,3,-124,246,2456,-345};
6
       sort(num.begin(),num.end());
7
       pair < vector < int >:: iterator, vector < int >:: iterator >
8
            minmaxElement=minmax_element(num.begin(),num.end());
       cout << "sorted vector :\n";</pre>
9
10
       for(auto x:num) cout<<x<"</pre>
       cout <<endl:
11
12
       cout << * minmaxElement . first << "
            "<<*minmaxElement.second<<endl;
13 }
```