Contents

1 Basic

1.1 ascii

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

1.2 limits

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

2 STL

2.1 priority_queue

```
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。
宣告：
    priority_queue <int> pq;
把元素 x 加進 priority_queue：
    pq.push(x);
讀取優先權最高的值：
    x = pq.top();
    pq.pop();                   //讀取後刪除
判斷是否為空的priority_queue：
    pq.empty()                  //回傳true
    pq.size()                   //回傳0
如需改變priority_queue的優先權定義：
    priority_queue<T> pq;       //預設由大到小
    priority_queue<T, vector<T>, greater<T> > pq;
                                //改成由小到大
    priority_queue<T, vector<T>, cmp> pq;   //cmp
```

2.2 map

```
map：存放 key-value pairs 的映射資料結構，會按 key
    由小到大排序。
元素存取
operator[]：存取指定的[i]元素的資料

迭代器
begin()：回傳指向map頭部元素的迭代器
end()：回傳指向map末尾的迭代器
rbegin()：回傳一個指向map尾部的反向迭代器
rend()：回傳一個指向map頭部的反向迭代器

遍歷整個map時，利用iterator操作：
取key：it->first 或 (*it).first
取value：it->second 或 (*it).second

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

修改器
clear()：刪除所有元素
insert()：插入元素
erase()：刪除一個元素
swap()：交換兩個map

查找
count()：回傳指定元素出現的次數
find()：查找一個元素

//實作範例
#include <bits/stdc++.h>
```

```cpp
using namespace std;

int main(){

    //declaration container and iterator
    map<string, string> mp;
    map<string, string>::iterator iter;
    map<string, string>::reverse_iterator iter_r;

    //insert element
    mp.insert(pair<string, string>("r000",
        "student_zero"));

    mp["r123"] = "student_first";
    mp["r456"] = "student_second";

    //traversal
    for(iter = mp.begin(); iter != mp.end(); iter++)
        cout<<iter->first<<" "<<iter->second<<endl;
    for(iter_r = mp.rbegin(); iter_r != mp.rend();
        iter_r++)
        cout<<iter_r->first<<"
            "<<iter_r->second<<endl;

    //find and erase the element
    iter = mp.find("r123");
    mp.erase(iter);

    iter = mp.find("r123");

    if(iter != mp.end())
       cout<<"Find, the value is
           "<<iter->second<<endl;
    else
       cout<<"Do not Find"<<endl;

    return 0;
}

//map統計數字
#include<bits/stdc++.h>
using namespace std;

int main(){
  ios::sync_with_stdio(0),cin.tie(0);
  long long n,x;
  cin>>n;
  map <int,int> mp;
  while(n--){
    cin>>x;
    ++mp[x];
  }
  for(auto i:mp) cout<<i.first<<" "<<i.second<<endl;
}
```

2.3 unordered_map

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

2.4 set

```
set： 集合，去除重複的元素，資料由小到大排序。
宣告：
    set <int> st;
把元素 x 加進 set：
    st.insert(x);
檢查元素 x 是否存在 set 中：
    st.count(x);
刪除元素 x：
    st.erase(x);      // 可傳入值或iterator
清空集合中的所有元素：
    st.clear();
取值： 使用iterator
    x = *st.begin();  //
        set中的第一個元素(最小的元素)。
    x = *st.rbegin(); //
        set中的最後一個元素(最大的元素)。
判斷是否為空的set：
st.empty() 回傳true
st.size() 回傳零
常用來搭配的member function：
st.count(x);
auto it = st.find(x); // binary search, O(log(N))
auto it = st.lower_bound(x); // binary search,
    O(log(N))
auto it = st.upper_bound(x); // binary search,
    O(log(N))
【multiset】
與 set
    用法雷同，但會保留重複的元素，資料由小到大排序。
宣告：
multiset<int> st;
刪除資料：
st.erase(val); 會刪除所有值為 val 的元素。
st.erase(st.find(val)); 只刪除第一個值為 val 的元素。
```

2.5 multiset

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

3 sort

3.1 big number sort

```
while True:
  try:
    n = int(input())           # 有幾筆數字需要排序
    arr = []                   # 建立空串列
    for i in range(n):
      arr.append(int(input())) # 依序將數字存入串列
    arr.sort()                 # 串列排序
    for i in arr:
      print(i)              # 依序印出串列中每個項目
  except:
    break
```

3.2 bubble sort

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int n;
  cin>>n;
  int a[n], tmp;
  for(int i=0; i<n; i++) cin>>a[i];
  for(int i=n-1; i>0; i--) {
    for(int j=0; j<=i-1; j++) {
      if( a[j]>a[j+1]) {
        tmp=a[j];
```

```
        a[j]=a[j+1];
        a[j+1]=tmp;
      }
    }
  }
  for(int i=0; i<n; i++) cout<<a[i]<<" ";
}
```

4 math

4.1 prime factorization

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
  int n;
  while(true) {
    cin>>n;
    for(int x=2; x<=n; x++) {
      while(n%x==0) {
        cout<<x<<"*";
        n/=x;
      }
    }
    cout<<"\b \n";
  }
  system("pause");
  return 0;
}
```

5 algorithm

5.1 basic

```
min： 取最小值。
min(a, b)
min(list)
max： 取最大值。
max(a, b)
max(list)
min_element： 找尋最小元素
min_element(first, last)
max_element： 找尋最大元素
max_element(first, last)
sort： 排序，預設由小排到大。
sort(first, last)
sort(first, last, comp)： 可自行定義比較運算子 Comp 。
find： 尋找元素。
find(first, last, val)
lower_bound： 尋找第一個小於 x
    的元素位置，如果不存在，則回傳 last 。
lower_bound(first, last, val)
upper_bound： 尋找第一個大於 x
    的元素位置，如果不存在，則回傳 last 。
upper_bound(first, last, val)
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 。
next_permutation(first, last)
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 。
prev_permutation(first, last)
```

5.2 binarysearch

```cpp
#include <bits/stdc++.h>
using namespace std;

int binary_search(vector<int> &nums, int target) {
    int left=0, right=nums.size()-1;
    while(left<=right){
        int mid=(left+right)/2;
        if (nums[mid]>target) right=mid-1;
        else if(nums[mid]<target) left=mid+1;
        else return mid+1;
    }
    return 0;
}

int main() {
  int n, k, x;
  cin >> n >> k;
  int a[n];
  vector<int> v;
  for(int i=0 ; i<n ; i++){
    cin >> x;
    v.push_back(x);
  }
  for(int i=0 ; i<k ; i++) cin >> a[i];
  for(int i=0 ; i<k ; i++){
    cout << binary_search(v, a[i]) << endl;
  }
}

/*
input
5 5
1 3 4 7 9
3 1 9 7 -2
*/

/*
output
2
1
5
4
0
*/
```

5.3 prefix sum

```cpp
// 前綴和
// 陣列前n項的和。
// b[i] = a[0] + a[1] + a[2] + … + a[i]
// 區間和 [l, r]：b[r]-b[l-1] (要保留b[1]所以-1)

#include <bits/stdc++.h>
using namespace std;
int main(){
    int n;
    cin >> n;
    int a[n], b[n];
    for(int i=0; i<n; i++) cin >> a[i];
    b[0] = a[0];
    for(int i=1; i<n; i++) b[i] = b[i-1] + a[i];
    for(int i=0;i<n;i++) cout<<b[i]<<' ';
    cout<<'\n';
    int l, r;
    cin >> l >> r;
    cout << b[r] - b[l-1] ; //區間和
}


// 差分
// 用途：在區間 [l, r] 加上一個數字v。
// b[l] += v; //b[0~l] 加上v
// 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[l] 加上 v，b[r+1] 減去 v，
// 最後再從 0 跑到 n 使 b[i] += b[i-1]。
// 這樣一來，b[] 是一個在某區間加上v的前綴和。

#include <bits/stdc++.h>
using namespace std;
int a[1000], b[1000];
//a: 前綴和數列, b: 差分數列
int main(){
    int n, l, r, v;
    cin >> n;
    for(int i=1; i<=n; i++){
        cin >> a[i];
        b[i] = a[i] - a[i-1]; //建構差分數列
    }
    cin >> l >> r >> v;
    b[l] += v;
    b[r+1] -= v;

    for(int i=1; i<=n; i++){
        b[i] += b[i-1];
        cout << b[i] << ' ';
    }
}
```

6 graph

6.1 graph

```

#include<bits/stdc++.h>
using namespace std;

class Node {
public:
    int val;
    vector<Node*> children;

    Node() {}

    Node(int _val) {
        val = _val;
    }

    Node(int _val, vector<Node*> _children) {
        val = _val;
        children = _children;
    }
};

struct ListNode {
    int val;
    ListNode *next;
    ListNode() : val(0), next(nullptr) {}
    ListNode(int x) : val(x), next(nullptr) {}
    ListNode(int x, ListNode *next) : val(x),
        next(next) {}
};

struct TreeNode {
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode() : val(0), left(nullptr),
        right(nullptr) {}
    TreeNode(int x) : val(x), left(nullptr),
        right(nullptr) {}
    TreeNode(int x, TreeNode *left, TreeNode *right)
        : val(x), left(left), right(right) {}
};

class ListProblem {
    vector<int> nums={};
public:
    void solve() {
        return;
    }

    ListNode* buildList(int idx) {
        if(idx == nums.size()) return NULL;
        ListNode *current=new
            ListNode(nums[idx++],current->next);
        return current;
    }

    void deleteList(ListNode* root) {
        if(root == NULL) return;
        deleteList(root->next);
        delete root;
        return;
    }
};

class TreeProblem {
    int null = INT_MIN;
    vector<int> nums = {}, result;
public:
    void solve() {

        return;
    }

    TreeNode* buildBinaryTreeUsingDFS(int left, int
        right) {
        if((left > right) || (nums[(left+right)/2] ==
            null)) return NULL;
        int mid = (left+right)/2;
        TreeNode* current = new TreeNode(
            nums[mid],
            buildBinaryTreeUsingDFS(left,mid-1),
            buildBinaryTreeUsingDFS(mid+1,right));
        return current;
    }

    TreeNode* buildBinaryTreeUsingBFS() {
        int idx = 0;
        TreeNode* root = new TreeNode(nums[idx++]);
        queue<TreeNode*> q;
        q.push(root);
        while(idx < nums.size()) {
            if(nums[idx] != null) {
                TreeNode* left = new
                    TreeNode(nums[idx]);
                q.front()->left = left;
                q.push(left);
            }
            idx++;
            if((idx < nums.size()) && (nums[idx] !=
                null)) {
                TreeNode* right = new
                    TreeNode(nums[idx]);
                q.front()->right = right;
                q.push(right);
            }
            idx++;
            q.pop();
        }
        return root;
    }

    Node* buildNAryTree() {
        int idx = 2;
        Node *root = new Node(nums.front());
        queue<Node*> q;
        q.push(root);
        while(idx < nums.size()) {
            while((idx < nums.size()) && (nums[idx]
                != null)) {
                Node *current = new Node(nums[idx++]);
                q.front()->children.push_back(current);
```

```
                q.push(current);
            }
            idx++;
            q.pop();
        }
        return root;
    }

    void deleteBinaryTree(TreeNode* root) {
        if(root->left != NULL)
            deleteBinaryTree(root->left);
        if(root->right != NULL)
            deleteBinaryTree(root->right);
        delete root;
        return;
    }

    void deleteNAryTree(Node* root) {
        if(root == NULL) return;
        for(int i=0; i<root->children.size(); i++) {
            deleteNAryTree(root->children[i]);
            delete root->children[i];
        }
        delete root;
        return;
    }

    void inorderTraversal(TreeNode* root) {
        if(root == NULL) return;
        inorderTraversal(root->left);
        cout<<root->val<<' ';
        inorderTraversal(root->right);
        return;
    }
};

int main() {

    return 0;
}
```

7 Section2

7.1 thm

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