

Contents

1	Basic	
1.1	ascii	
1.2	limits	
2	字串	
2.1	最長迴文子字串	
3	STL	
3.1	priority_queue	
3.2	queue	
3.3	deque	
3.4	map	
3.5	unordered_map	
3.6	set	
3.7	multiset	
3.8	unordered_set	
3.9	單調隊列	
4	sort	
4.1	big number sort	
4.2	bubble sort	
5	math	
5.1	prime factorization	
5.2	快速冪	
6	algorithm	
6.1	basic	
6.2	binarysearch	
6.3	prefix sum	
6.4	差分	
6.5	greedy	
7	graph	
7.1	graph	
8	Section2	
8.1	thm	

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]
1	char	1	127 to -128
2	signed char	1	127 to -128
3	unsigned char	1	0 to 255
4	short	2	32767 to -32768
5	int	4	2147483647 to -2147483648
6	unsigned int	4	0 to 4294967295
7	long	4	2147483647 to -2147483648
8	unsigned long	4	0 to 18446744073709551615
9	long long	8	
10		8	
11		9223372036854775807 to -9223372036854775808	
12	double	8	1.79769e+308 to 2.22507e-308
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
16	string	32	

2 字串

2.1 最長迴文子字串

```

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

```

3 STL

3.1 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
2 宣告：
   priority_queue <int> pq;
3 把元素 x 加進 priority_queue：
   pq.push(x);
4 讀取優先權最高的值：
   x = pq.top();
5 判斷是否為空的priority_queue：
   pq.empty()           //回傳true
   pq.size()            //回傳0
6 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;
   priority_queue<T, vector<T>, cmp> pq; //cmp

```

3.2 queue

```

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

```

3.3 deque

```

1 deque 是 C++ 標準模板函式庫 (Standard Template
   Library, STL)
2 中的雙向佇列容器 (Double-ended Queue)，跟 vector
   相似，
3 不過在 vector
   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)。同樣地，
也能在我們需要儲存更多元素的時候自動擴展空間，
讓我們不必煩惱佇列長度的問題。

```

7 dq.push_back() //在 deque 的最尾端新增元素
8 dq.push_front() //在 deque 的開頭新增元素
9 dq.pop_back() //移除 deque 最尾端的元素
10 dq.pop_front() //移除 deque 最開頭的元素
11 dq.back() //取出 deque 最尾端的元素
12 dq.front() //回傳 deque 最開頭的元素
13 dq.insert()
14 dq.insert(position, n, val)
   position: 插入元素的 index 值
   n: 元素插入次數
   val: 插入的元素值
15 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。
16 dq.clear() //清空整個 deque 佇列。
17 dq.size() //檢查 deque 的尺寸
18 dq.empty() //如果 deque 佇列為空返回 1；
   若是存在任何元素，則返回 0
19 dq.begin() //返回一個指向 deque 開頭的迭代器
20 dq.end() //指向 deque 結尾，
   不是最後一個元素，
   而是最後一個元素的下一個位置

```

3.4 map

```

1 map：存放 key-value pairs 的映射資料結構，會按 key
   由小到大排序。
2 元素存取
3 operator[]：存取指定的[i]元素的資料
4
5 迭代器
6 begin()：回傳指向map頭部元素的迭代器
7 end()：回傳指向map末尾的迭代器
8 rbegin()：回傳一個指向map尾部的反向迭代器
9 rend()：回傳一個指向map頭部的反向迭代器
10
11 遍歷整個map時，利用iterator操作：
12 取key：it->first 或 (*it).first
13 取value：it->second 或 (*it).second
14
15 容量
16 empty()：檢查容器是否為空，空則回傳true
17 size()：回傳元素數量
18 max_size()：回傳可以容納的最大元素個數
19
20 修改器
21 clear()：刪除所有元素
22 insert()：插入元素
23 erase()：刪除一個元素
24 swap()：交換兩個map
25
26 查找
27 count()：回傳指定元素出現的次數
28 find()：查找一個元素
29
30 //實作範例
31 #include <bits/stdc++.h>
32 using namespace std;
33
34 int main(){
35
36     //declaration container and iterator
37     map<string, string> mp;
38     map<string, string>::iterator iter;
39     map<string, string>::reverse_iterator iter_r;
40
41     //insert element

```

```

42 mp.insert(pair<string, string>("r000",
43     "student_zero"));
44 mp["r123"] = "student_first";
45 mp["r456"] = "student_second";
46
47 //traversal
48 for(iter = mp.begin(); iter != mp.end(); iter++)
49     cout<<iter->first<<" "<<iter->second<<endl;
50 for(iter_r = mp.rbegin(); iter_r != mp.rend();
51     iter_r++)
52     cout<<iter_r->first<<"
53         "<<iter_r->second<<endl;
54
55 //find and erase the element
56 iter = mp.find("r123");
57 mp.erase(iter);
58
59 iter = mp.find("r123");
60
61 if(iter != mp.end())
62     cout<<"Find, the value is
63         "<<iter->second<<endl;
64 else
65     cout<<"Do not Find"<<endl;
66
67 return 0;
68 }
69
70 //map統計數字
71 #include<bits/stdc++.h>
72 using namespace std;
73
74 int main(){
75     ios::sync_with_stdio(0),cin.tie(0);
76     long long n,x;
77     cin>>n;
78     map <int,int> mp;
79     while(n--){
80         cin>>x;
81         ++mp[x];
82     }
83     for(auto i:mp) cout<<i.first<<" "<<i.second<<endl;
84 }

```

3.5 unordered_map

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

3.6 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 把元素 x 加進 set：
 5 st.insert(x);
 6 檢查元素 x 是否存在 set 中：
 7 st.count(x);
 8 刪除元素 x：
 9 st.erase(x); // 可傳入值或 iterator
 10 清空集合中的所有元素：
 11 st.clear();
 12 取值：使用 iterator
 13 x = *st.begin();
 14 // set 中的第一個元素(最小的元素)。
 15 x = *st.rbegin();
 16 // set 中的最後一個元素(最大的元素)。
 17 判斷是否為空的 set：
 18 st.empty() 回傳 true
 19 st.size() 回傳零

20 常用來搭配的 member function：
 21 st.count(x);
 22 auto it = st.find(x);
 23 // binary search, $O(\log(N))$
 24 auto it = st.lower_bound(x);
 25 // binary search, $O(\log(N))$
 26 auto it = st.upper_bound(x);
 27 // binary search, $O(\log(N))$

3.7 multiset

1 與 set 用法雷同，但會保留重複的元素，
 資料由小到大排序。
 2 宣告：
 3 multiset<int> st;
 4 刪除資料：
 5 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。

3.8 unordered_set

1 unordered_set 的實作方式通常是用雜湊表(hash table)，
 2 資料插入和查詢的時間複雜度很低，為常數級別 $O(1)$ ，
 3 相對的代價是消耗較多的記憶體，空間複雜度較高，
 4 無自動排序功能。
 5
 6 初始化
 7 unordered_set<int> myunordered_set{1, 2, 3, 4, 5};
 8
 9 陣列初始化
 10 int arr[] = {1, 2, 3, 4, 5};
 11 unordered_set<int> myunordered_set(arr, arr+5);
 12
 13 插入元素
 14 unordered_set<int> myunordered_set;
 15 myunordered_set.insert(1);
 16
 17 迴圈遍歷 unordered_set 容器
 18 #include <iostream>
 19 #include <unordered_set>
 20 using namespace std;
 21
 22 int main() {
 23 unordered_set<int> myunordered_set = {3, 1};
 24 myunordered_set.insert(2);
 25 myunordered_set.insert(5);
 26 myunordered_set.insert(4);
 27 myunordered_set.insert(5);
 28 myunordered_set.insert(4);
 29
 30 for (const auto &s : myunordered_set) {
 31 cout << s << " ";
 32 }
 33 cout << "\n";
 34
 35 return 0;
 36 }
 37
 38 /*
 39 output
 40 4 5 2 1 3
 41 */
 42
 43 unordered_set 刪除指定元素
 44 #include <iostream>
 45 #include <unordered_set>
 46
 47 int main() {
 48 unordered_set<int> myunordered_set{2, 4, 6, 8};
 49
 50 myunordered_set.erase(2);

```

51     for (const auto &s : myunordered_set) {
52         cout << s << " ";
53     }
54     cout << "\n";
55     return 0;
56 }
57 /*
58 output
59 8 6 4
60 */
61
62 清空 unordered_set 元素
63 unordered_set<int> myunordered_set;
64 myunordered_set.insert(1);
65 myunordered_set.clear();
66
67 unordered_set 判斷元素是否存在
68 unordered_set<int> myunordered_set;
69 myunordered_set.insert(2);
70 myunordered_set.insert(4);
71 myunordered_set.insert(6);
72 cout << myunordered_set.count(4) << "\n"; // 1
73 cout << myunordered_set.count(8) << "\n"; // 0
74
75 判斷 unordered_set 容器是否為空
76 #include <iostream>
77 #include <unordered_set>
78
79 int main() {
80     unordered_set<int> myunordered_set;
81     myunordered_set.clear();
82
83     if (myunordered_set.empty()) {
84         cout << "empty\n";
85     } else {
86         cout << "not empty, size is "<<
87             myunordered_set.size() << "\n";
88     }
89
90     return 0;
91 }

```

3.9 單調隊列

```

1 //單調隊列
2 "如果一個選手比你小還比你強，你就可以退役了。"--單調隊列
3
4 example 1
5
6 給出一個長度為 n 的數組，
7 編程輸出每 k 個連續的數中的最大值和最小值。
8 //寫法一
9 #include <bits/stdc++.h>
10 #define maxn 1000100
11 using namespace std;
12 int q[maxn], a[maxn];
13 int n, k;
14
15 void getmin() {
16     // 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     for (int i = 1; i < k; i++) {
19         while (head <= tail && a[q[tail]] >= a[i])
20             tail--;
21         q[++tail] = i;
22     }
23     for (int i = k; i <= n; i++) {
24         while (head <= tail && a[q[tail]] >= a[i])
25             tail--;
26         q[++tail] = i;
27         while (q[head] <= i - k) head++;
28         cout << a[q[head]] << " ";
29     }
30 }

```

```

29
30 void getmax() { // 和上面同理
31     int head = 0, tail = 0;
32     for (int i = 1; i < k; i++) {
33         while (head <= tail && a[q[tail]] <= a[i])
34             tail--;
35         q[++tail] = i;
36     }
37     for (int i = k; i <= n; i++) {
38         while (head <= tail && a[q[tail]] <= a[i])
39             tail--;
40         q[++tail] = i;
41         while (q[head] <= i - k) head++;
42         cout << a[q[head]] << " ";
43     }
44 }
45
46 int main() {
47     cin >> n >> k; // 每 k 個連續的數
48     for (int i = 1; i <= n; i++) cin >> a[i];
49     getmin();
50     getmax();
51     cout << "\n";
52     return 0;
53 }
54
55 //寫法2
56 #include <iostream>
57 #include <cstring>
58 #include <deque>
59 using namespace std;
60 int a[1000005];
61
62 int main() {
63     ios_base::sync_with_stdio(0);
64     int n, k;
65     while (cin >> n >> k) {
66         for (int i = 0; i < n; i++) cin >> a[i];
67         deque<int> dq;
68         for (int i = 0; i < n; i++) {
69             while (dq.size() && dq.front() <= i - k)
70                 dq.pop_front();
71             while (dq.size() && a[dq.back()] > a[i])
72                 dq.pop_back();
73             dq.push_back(i);
74             if (i == k - 1) cout << a[dq.front()];
75             if (i > k - 1) cout << ' ' << a[dq.front()];
76         }
77         if (k > n) cout << a[dq.front()];
78         cout << "\n";
79         while (dq.size()) dq.pop_back();
80         for (int i = 0; i < n; i++) {
81             while (dq.size() && dq.front() <= i - k)
82                 dq.pop_front();
83             while (dq.size() && a[dq.back()] < a[i])
84                 dq.pop_back();
85             dq.push_back(i);
86             if (i == k - 1) cout << a[dq.front()];
87             if (i > k - 1) cout << ' ' << a[dq.front()];
88         }
89         if (k > n) cout << a[dq.front()];
90         cout << "\n";
91     }
92     return 0;
93 }
94
95 example 2
96
97 一個含有 n 項的數列，求出每一項前的 m
98 個數到它這個區間內的最小值。
99 若前面的數不足 m 項則從第 1
100 個數開始，若前面沒有數則輸出 0
101
102 #include <bits/stdc++.h>
103 using namespace std;

```

```

102 #define re register int
103 #define INF 0x3f3f3f3f
104 #define ll long long
105 #define maxn 2000009
106 #define maxm
107 inline ll read() {
108     ll x=0,f=1;
109     char ch=getchar();
110     while(ch<'0' || ch>'9'){
111         if(ch=='-') f=-1;
112         ch=getchar();
113     }
114     while(ch>='0' && ch<='9'){
115         x=(x<<1)+(x<<3)+(ll)(ch-'0');
116         ch=getchar();
117     }
118     return x*f;
119 }
120 int n,m,k,tot,head,tail;
121 int a[maxn],q[maxn];
122 int main() {
123     n=read(), m=read();
124     for(int i=1;i<=n;i++) a[i]=read();
125     head=1,tail=0; //起始位置為1
    //因為插入是q[++tail]所以要初始化為0
126     for(int i=1;i<=n;i++)
127         //每次隊首的元素就是當前的答案
128     {
129         cout<<a[q[head]]<<endl;
130         while(i-q[head]+1>m && head<=tail) //維護隊首
131             head++;
132         while(a[i]<a[q[tail]] && head<=tail) //維護隊尾
133             tail--;
134         q[++tail]=i;
135     }
136     return 0;
137 }

```

4 sort

4.1 big number sort

```

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

```

4.2 bubble sort

```

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

```

```

14                 a[j+1]=tmp;
15             }
16         }
17     }
18     for(int i=0; i<n; i++) cout<<a[i]<<" ";
19 }

```

5 math

5.1 prime factorization

```

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

```

5.2 快速幂

```

1 計算a^b
2 #include <iostream>
3 #define ll long long
4 using namespace std;
5
6 const ll MOD = 1000000007;
7 ll fp(ll a, ll b) {
8     int ans = 1;
9     while(b > 0) {
10         if(b & 1) ans = ans * a % MOD;
11         a = a * a % MOD;
12         b >>= 1;
13     }
14     return ans;
15 }
16
17 int main() {
18     int a, b;
19     cin>>a>>b;
20     cout<<fp(a,b);
21 }

```

6 algorithm

6.1 basic

```

1 min： 取最小值。
2 min(a, b)
3 min(list)
4 max： 取最大值。
5 max(a, b)
6 max(list)
7 min_element： 找尋最小元素
8 min_element(first, last)
9 max_element： 找尋最大元素
10 max_element(first, last)

```

```

11 sort: 排序，預設由小排到大。
12 sort(first, last)
13 sort(first, last, comp): 可自行定義比較運算子 Comp。
14 find: 尋找元素。
15 find(first, last, val)
16 lower_bound: 尋找第一個小於 x
    的元素位置，如果不存在，則回傳 last。
17 lower_bound(first, last, val)
18 upper_bound: 尋找第一個大於 x
    的元素位置，如果不存在，則回傳 last。
19 upper_bound(first, last, val)
20 next_permutation:
    將序列順序轉換成下一個字典序，如果存在回傳 true
    ，反之回傳 false。
21 next_permutation(first, last)
22 prev_permutation:
    將序列順序轉換成上一個字典序，如果存在回傳 true
    ，反之回傳 false。
23 prev_permutation(first, last)

```

6.2 binarysearch

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int binary_search(vector<int> &nums, int target) {
5     int left=0, right=nums.size()-1;
6     while(left<=right){
7         int mid=(left+right)/2;
8         if (nums[mid]>target) right=mid-1;
9         else if(nums[mid]<target) left=mid+1;
10        else return mid+1;
11    }
12    return 0;
13 }
14
15 int main() {
16     int n, k, x;
17     cin >> n >> k;
18     int a[n];
19     vector<int> v;
20     for(int i=0 ; i<n ; i++){
21         cin >> x;
22         v.push_back(x);
23     }
24     for(int i=0 ; i<k ; i++) cin >> a[i];
25     for(int i=0 ; i<k ; i++){
26         cout << binary_search(v, a[i]) << endl;
27     }
28 }
29
30 lower_bound(a, a + n, k); //最左邊 ≥ k 的位置
31 upper_bound(a, a + n, k); //最左邊 > k 的位置
32 upper_bound(a, a + n, k) - 1; //最右邊 ≤ k 的位置
33 lower_bound(a, a + n, k) - 1; //最右邊 < k 的位置
34 (lower_bound, upper_bound) //等於 k 的範圍
35 equal_range(a, a+n, k);
36
37 /*
38 input
39 5 5
40 1 3 4 7 9
41 3 1 9 7 -2
42 */
43
44 /*
45 output
46 2
47 1
48 5
49 4
50 0
51 */

```

6.3 prefix sum

```

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

```

6.4 差分

```

1 // 差分
2 用途：在區間 [l, r] 加上一個數字v。
3 b[l] += v; (b[0~l] 加上v)
4 b[r+1] -= v; (b[r+1~n] 減去v (b[r] 仍保留v) )
5 給的 a[] 是前綴和數列，建構 b[]，
6 因為 a[i] = b[0] + b[1] + b[2] + ... + b[i]，
7 所以 b[i] = a[i] - a[i-1]。
8 在 b[l] 加上 v，b[r+1] 減去 v，
9 最後再從 0 跑到 n 使 b[i] += b[i-1]。
10 這樣一來，b[] 是一個在某區間加上v的前綴和。
11
12 #include <bits/stdc++.h>
13 using namespace std;
14 int a[1000], b[1000];
15 //a: 前綴和數列，b: 差分數列
16 int main(){
17     int n, l, r, v;
18     cin >> n;
19     for(int i=1; i<=n; i++){
20         cin >> a[i];
21         b[i] = a[i] - a[i-1]; //建構差分數列
22     }
23     cin >> l >> r >> v;
24     b[l] += v;
25     b[r+1] -= v;
26
27     for(int i=1; i<=n; i++){
28         b[i] += b[i-1];
29         cout << b[i] << ' ';
30     }
31 }

```

6.5 greedy

```

1 //貪心
2 貪心演算法的核心為，
3 採取在目前狀態下最好或最佳（即最有利）的選擇。
4 貪心演算法雖然能獲得當前最佳解，
5 但不保證能獲得最後（全域）最佳解，
6 提出想法後可以先試圖尋找有沒有能推翻原本的想法的反例，
7 確認無誤再實作。
8
9 //problem
10 有一個 N×1 的稻田，有些稻田現在有種植作物，
11 為了避免被動物破壞，需要放置稻草人，

```

```

12 稻草人可以保護該塊稻田和左右兩塊稻田，
13 請問最少需要多少稻草人才能保護所有稻田？
14
15 //solutoin
16 從左到右掃描稻田，如果第 i 塊稻田有作物，
17 就把稻草人放到第 i+1 塊稻田，
18 這樣能保護第 i,i+1,i+2 塊稻田，
19 接著從第 i+3 塊稻田繼續掃描。
20
21 //code
22 #include <bits/stdc++.h>
23 using namespace std;
24 int main(){
25     string s;
26     int i, n, t, tc = 1;
27     cin >> t;
28     while (t--){
29         cin >> n >> s;
30         int nc = 0;
31         for (i = 0; i < n; i++){
32             if (s[i] == '.') i += 2, nc++;
33         }
34         cout << "Case " << tc++ << ": " << nc << endl;
35     }
36
37 //problem
38 給定 N 個數，每次將兩個數 a,b 合併成 a+b，
39 只到最後只剩一個數，合併成本為兩數和，問最小合併成本為多少
40

```

7 graph

7.1 graph

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  class Node {
5  public:
6      int val;
7      vector<Node*> children;
8
9      Node() {}
10
11     Node(int _val) {
12         val = _val;
13     }
14
15     Node(int _val, vector<Node*> _children) {
16         val = _val;
17         children = _children;
18     }
19 };
20
21 struct ListNode {
22     int val;
23     ListNode *next;
24     ListNode() : val(0), next(nullptr) {}
25     ListNode(int x) : val(x), next(nullptr) {}
26     ListNode(int x, ListNode *next) : val(x),
27         next(next) {}
28 };
29 struct TreeNode {
30     int val;
31     TreeNode *left;
32     TreeNode *right;
33     TreeNode() : val(0), left(nullptr),
34         right(nullptr) {}
35     TreeNode(int x) : val(x), left(nullptr),
36         right(nullptr) {}
37     TreeNode(int x, TreeNode *left, TreeNode *right)
38         : val(x), left(left), right(right) {}
39 };

```

```

36 };
37
38 class ListProblem {
39     vector<int> nums={};
40 public:
41     void solve() {
42         return;
43     }
44
45     ListNode* buildList(int idx) {
46         if(idx == nums.size()) return NULL;
47         ListNode *current=new
48             ListNode(nums[idx++],current->next);
49         return current;
50     }
51
52     void deleteList(ListNode* root) {
53         if(root == NULL) return;
54         deleteList(root->next);
55         delete root;
56         return;
57     }
58
59 class TreeProblem {
60     int null = INT_MIN;
61     vector<int> nums = {}, result;
62 public:
63     void solve() {
64
65         return;
66     }
67
68     TreeNode* buildBinaryTreeUsingDFS(int left, int
69         right) {
70         if((left > right) || (nums[(left+right)/2] ==
71             null)) return NULL;
72         int mid = (left+right)/2;
73         TreeNode* current = new TreeNode(
74             nums[mid],
75             buildBinaryTreeUsingDFS(left,mid-1),
76             buildBinaryTreeUsingDFS(mid+1,right));
77         return current;
78     }
79
80     TreeNode* buildBinaryTreeUsingBFS() {
81         int idx = 0;
82         TreeNode* root = new TreeNode(nums[idx++]);
83         queue<TreeNode*> q;
84         q.push(root);
85         while(idx < nums.size()) {
86             if(nums[idx] != null) {
87                 TreeNode* left = new
88                     TreeNode(nums[idx]);
89                 q.front()->left = left;
90                 q.push(left);
91             }
92             idx++;
93             if((idx < nums.size()) && (nums[idx] !=
94                 null)) {
95                 TreeNode* right = new
96                     TreeNode(nums[idx]);
97                 q.front()->right = right;
98                 q.push(right);
99             }
100             idx++;
101             q.pop();
102         }
103         return root;
104     }
105
106     Node* buildNaryTree() {
107         int idx = 2;
108         Node *root = new Node(nums.front());
109         queue<Node*> q;
110         q.push(root);
111         while(idx < nums.size()) {

```



```

107         while((idx < nums.size()) && (nums[idx]
108             != null)) {
109             Node *current = new Node(nums[idx++]);
110             q.front()->children.push_back(current);
111             q.push(current);
112         }
113         idx++;
114         q.pop();
115     }
116     return root;
117 }
118
119 void deleteBinaryTree(TreeNode* root) {
120     if(root->left != NULL)
121         deleteBinaryTree(root->left);
122     if(root->right != NULL)
123         deleteBinaryTree(root->right);
124     delete root;
125     return;
126 }
127
128 void deleteNaryTree(Node* root) {
129     if(root == NULL) return;
130     for(int i=0; i<root->children.size(); i++) {
131         deleteNaryTree(root->children[i]);
132         delete root->children[i];
133     }
134     delete root;
135     return;
136 }
137
138 void inorderTraversal(TreeNode* root) {
139     if(root == NULL) return;
140     inorderTraversal(root->left);
141     cout<<root->val<<' ';
142     inorderTraversal(root->right);
143     return;
144 }
145
146 };
147
148 int main() {
149     return 0;
150 }

```

8 Section2

8.1 thm

- 中文測試

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