

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	
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	9223372036854775807 to -9223372036854775808
11	double	8	1.79769e+308 to 2.22507e-308
12	long double	16	1.18973e+4932 to 3.3621e-4932
13	float	4	3.40282e+38 to 1.17549e-38
14	unsigned long long	8	0 to 18446744073709551615
15	string	32	

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：
3   priority_queue <int> pq;
4 把元素 x 加進 priority_queue：
5   pq.push(x);
6 讀取優先權最高的值：
7   x = pq.top();
8   pq.pop();           //讀取後刪除
9 判斷是否為空的priority_queue：
10  pq.empty()           //回傳true
11  pq.size()            //回傳0
12 如需改變priority_queue的優先權定義：
13  priority_queue<T> pq; //預設由大到小
14  priority_queue<T, vector<T>, greater<T> > pq;
15                          //改成由小到大
16  priority_queue<T, vector<T>, cmp> pq; //cmp

```

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：
4   queue <int> q;
5 把元素 x 加進 queue：
6   q.push(x);
7 取值：
8   x = q.front(); //頭
9   x = q.back();  //尾
10 移除已經讀取的值：
11   q.pop();
12 判斷是否為空的queue：
13   q.empty() 回傳true
14   q.size() 回傳零
15
16 #include <iostream>
17 #include <queue>
18 using namespace std;
19
20 int main() {
21     int n;
22     while (cin >> n){
23         if (n == 0) break;
24         queue <int> q;
25         for (int i = 0; i < n; i++){
26             q.push(i+1);
27         }
28         cout << "Discarded cards:";
29         for (int i = 0; i < n-1; i++){
30             if (i != 0) cout << ', ';
31             cout << ' ' << q.front();
32             q.pop();
33             q.push(q.front());
34             q.pop();
35         }
36         cout << endl << "Remaining card: " <<
            q.front() << endl;
37     }
38 }

```

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)
15     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 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。
19 dq.clear() //清空整個 deque 佇列。
20 dq.size() //檢查 deque 的尺寸
21 dq.empty() //如果 deque 佇列為空返回 1；
   若是存在任何元素，則返回 0
22 dq.begin() //返回一個指向 deque 開頭的迭代器
23 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
25

```

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

```

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); // binary search, O(log(N))
23 auto it = st.lower_bound(x);
24                                     // binary search, O(log(N))
25 auto it = st.upper_bound(x);
26                                     // 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 初始化
2 unordered_set<int> myunordered_set{1, 2, 3, 4, 5};
3
4 陣列初始化
5 int arr[] = {1, 2, 3, 4, 5};
6 unordered_set<int> myunordered_set(arr, arr+5);
7
8 插入元素
9 unordered_set<int> myunordered_set;
10 myunordered_set.insert(1);
11
12 迴圈遍歷 unordered_set 容器
13 #include <iostream>
14 #include <unordered_set>
15 using namespace std;
16
17 int main() {
18     unordered_set<int> myunordered_set = {3, 1};
19     myunordered_set.insert(2);
20     myunordered_set.insert(5);
21     myunordered_set.insert(4);
22     myunordered_set.insert(5);
23     myunordered_set.insert(4);
24
25     for (const auto &s : myunordered_set) {
26         cout << s << " ";
27     }
28     cout << "\n";
29
30     return 0;
31 }
32
33 /*
34 output
35 4 5 2 1 3
36 */
37
38 unordered_set 刪除指定元素
39 #include <iostream>
40 #include <unordered_set>
41
42 int main() {
43     unordered_set<int> myunordered_set{2, 4, 6, 8};
44
45     myunordered_set.erase(2);
46     for (const auto &s : myunordered_set) {
47         cout << s << " ";
48     }
49     cout << "\n";
50
51     return 0;

```

```

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

```

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
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]) tail--;
20         q[++tail] = i;
21     }
22     for (int i = k; i <= n; i++) {
23         while (head <= tail && a[q[tail]] >= a[i]) tail--;
24         q[++tail] = i;
25         while (q[head] <= i - k) head++;
26         printf("%d ", a[q[head]]);
27     }
28 }
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]) tail--;
34         q[++tail] = i;
35     }
36     for (int i = k; i <= n; i++) {

```

```

37     while (head <= tail && a[q[tail]] <= a[i]) tail--;
38     q[++tail] = i;
39     while (q[head] <= i - k) head++;
40     cout<<a[q[head]]<<" ";
41 }
42 }
43
44 int main() {
45     cin>>n>>k; //每k個連續的數
46     for (int i = 1; i <= n; i++) cin>>a[i];
47     getmin();
48     cout<<'\n';
49     getmax();
50     cout<<'\n';
51     return 0;
52 }
53
54 example 2
55
56 一個含有 n 項的數列，求出每一項前的 m
57 個數到它這個區間內的最小值。
58 若前面的數不足 m 項則從第 1
59 個數開始，若前面沒有數則輸出 0
60
61 #include<bits/stdc++.h>
62 using namespace std;
63 #define re register int
64 #define INF 0x3f3f3f3f
65 #define ll long long
66 #define maxn 2000009
67 #define maxm
68 inline ll read() {
69     ll x=0,f=1;char ch=getchar();
70     while(ch<'0' || ch>'9'){
71         if(ch=='-') f=-1;
72         ch=getchar();
73     }
74     while(ch>='0' && ch<='9'){
75         x=(x<<1)+(x<<3)+(ll)(ch-'0');
76         ch=getchar();
77     }
78     return x*f;
79 }
80 int n,m,k,tot,head,tail;
81 int a[maxn],q[maxn];
82 int main() {
83     n=read(), m=read();
84     for(int i=1;i<=n;i++) a[i]=read();
85     head=1,tail=0; //起始位置為1
86     //因為插入是q[++tail]所以要初始化為0
87     for(int i=1;i<=n;i++) //每次隊首的元素就是當前的答案
88     {
89         printf("%d\n",a[q[head]]);
90         while(i-q[head]+1>m&&head<=tail) //維護隊首
91             head++;
92         while(a[i]<a[q[tail]]&&head<=tail) //維護隊尾
93             tail--;
94         q[++tail]=i;
95     }
96     return 0;
97 }

```

4 sort

4.1 big number sort

```

1 while True:
2     try:
3         n = int(input())
4         arr = []
5         for i in range(n):

```

有幾筆數字需要排序
建立空串列

```

6 |     arr.append(int(input())) # 依序將數字存入串列
7 |     arr.sort()              # 串列排序
8 |     for i in arr:
9 |         print(i)            # 依序印出串列中每個項目
10 | except:
11 |     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 快速冪

```

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 }

```

7 graph

7.1 graph

```

1
2 #include<bits/stdc++.h>
3 using namespace std;
4
5 class Node {
6 public:
7     int val;
8     vector<Node*> children;
9
10    Node() {}
11
12    Node(int _val) {
13        val = _val;
14    }
15
16    Node(int _val, vector<Node*> _children) {
17        val = _val;
18        children = _children;
19    }
20 };
21
22 struct ListNode {
23     int val;
24     ListNode *next;
25     ListNode() : val(0), next(nullptr) {}
26     ListNode(int x) : val(x), next(nullptr) {}
27     ListNode(int x, ListNode *next) : val(x),
28         next(next) {}
29 };
30
31 struct TreeNode {
32     int val;
33     TreeNode *left;
34     TreeNode *right;
35     TreeNode() : val(0), left(nullptr),
36         right(nullptr) {}
37     TreeNode(int x) : val(x), left(nullptr),
38         right(nullptr) {}
39     TreeNode(int x, TreeNode *left, TreeNode *right)
40         : val(x), left(left), right(right) {}
41 };
42
43 class ListProblem {
44     vector<int> nums={};
45 public:
46     void solve() {
47         return;
48     }
49
50     ListNode* buildList(int idx) {
51         if(idx == nums.size()) return NULL;
52         ListNode *current=new
53             ListNode(nums[idx++],current->next);
54         return current;
55     }
56
57     void deleteList(ListNode* root) {
58         if(root == NULL) return;
59         deleteList(root->next);
60         delete root;
61         return;
62     }
63 }

```

```

58 };
59
60 class TreeProblem {
61     int null = INT_MIN;
62     vector<int> nums = {}, result;
63 public:
64     void solve() {
65
66         return;
67     }
68
69     TreeNode* buildBinaryTreeUsingDFS(int left, int
70         right) {
71         if((left > right) || (nums[(left+right)/2] ==
72             null)) return NULL;
73         int mid = (left+right)/2;
74         TreeNode* current = new TreeNode(
75             nums[mid],
76             buildBinaryTreeUsingDFS(left, mid-1),
77             buildBinaryTreeUsingDFS(mid+1, right));
78         return current;
79     }
80
81     TreeNode* buildBinaryTreeUsingBFS() {
82         int idx = 0;
83         TreeNode* root = new TreeNode(nums[idx++]);
84         queue<TreeNode*> q;
85         q.push(root);
86         while(idx < nums.size()) {
87             if(nums[idx] != null) {
88                 TreeNode* left = new
89                     TreeNode(nums[idx]);
90                 q.front()->left = left;
91                 q.push(left);
92             }
93             idx++;
94             if((idx < nums.size()) && (nums[idx] !=
95                 null)) {
96                 TreeNode* right = new
97                     TreeNode(nums[idx]);
98                 q.front()->right = right;
99                 q.push(right);
100             }
101             idx++;
102             q.pop();
103         }
104         return root;
105     }
106
107     Node* buildNArYTree() {
108         int idx = 2;
109         Node *root = new Node(nums.front());
110         queue<Node*> q;
111         q.push(root);
112         while(idx < nums.size()) {
113             while((idx < nums.size()) && (nums[idx]
114                 != null)) {
115                 Node *current = new Node(nums[idx++]);
116                 q.front()->children.push_back(current);
117                 q.push(current);
118             }
119             idx++;
120             q.pop();
121         }
122         return root;
123     }
124
125     void deleteBinaryTree(TreeNode* root) {
126         if(root->left != NULL)
127             deleteBinaryTree(root->left);
128         if(root->right != NULL)
129             deleteBinaryTree(root->right);
130         delete root;
131         return;
132     }
133
134     void deleteNArYTree(Node* root) {

```

```

127         if(root == NULL) return;
128         for(int i=0; i<root->children.size(); i++) {
129             deleteNArYTree(root->children[i]);
130             delete root->children[i];
131         }
132         delete root;
133         return;
134     }
135
136     void inorderTraversal(TreeNode* root) {
137         if(root == NULL) return;
138         inorderTraversal(root->left);
139         cout<<root->val<< ' ';
140         inorderTraversal(root->right);
141         return;
142     }
143 };
144
145 int main() {
146
147     return 0;
148 }

```

8 Section2

8.1 thm

- 中文測試

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