**★★★★☆**

**題組：Contest Set Archive with Online Judge**

**題號：10534 Wavio Sequence**

**解題者：陳冠綸**

**解題日期：2021年5月27日**

**題意：** 給定一數列，需找出此數列中子序列，並且此子序列項數為2\*n-1項(n為正整數)，第一項到第n項為嚴格遞增，第n項到最後一項為嚴格遞減，遞增的項數和遞減的項數要一樣，找出符合這些的子序列中最長的序列，並輸出其長度。

**題意範例：**

**Case1:**

**10**

**1 2 3 4 5 4 3 2 1 10 9**

**19**

**1 2 3 2 1 2 3 4 3 2 1 5 4 1 2 3 2 2 1 9**

**5**

**1 2 3 4 5 1**

**解法：**

找出LIS(Longest increasing subseqence)和LDS並建表，比較LIS和LDS求出的長度，找出最長的長度，再將最大值乘2減1

**解法範例：**

方法一 :

以1 4 2 3 4 2 1為範例

分別以各個數字為結尾，找出以此數字為結尾的長度

因為最短長度是以自己為結尾，所以Len初始值設為1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 4 | 2 | 3 | 4 | 2 | 1 |
| Len | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

因為4>1，4可接在1後面，形成遞增子序列{1,4}

長度為2，Len更新為2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 4 | 2 | 3 | 4 | 2 | 1 |
| Len | 1 | 2 | 1 | 1 | 1 | 1 | 1 |

因為2>1，2可以接在1後面，形成遞增子序列{1,2}

長度為2，Len更新為2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 4 | 2 | 3 | 4 | 2 | 1 |
| Len | 1 | 2 | 2 | 1 | 1 | 1 | 1 |

因為2<4，不可接在4後面，長度為1，不需更新

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 4 | 2 | 3 | 4 | 2 | 1 |
| Len | 1 | 2 | 2 | 1 | 1 | 1 | 1 |

因為3>1，3可以接在1後面，形成遞增子序列{1,3}

長度為2，Len更新為2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 4 | 2 | 3 | 4 | 2 | 1 |
| Len | 1 | 2 | 2 | 2 | 1 | 1 | 1 |

因為3<4，不可接在4後面，長度為1，不需更新

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 4 | 2 | 3 | 4 | 2 | 1 |
| Len | 1 | 2 | 2 | 2 | 1 | 1 | 1 |

因為3>2，3可以接在2後面，形成遞增子序列{1,2,3}

長度為3，Len更新為3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 4 | 2 | 3 | 4 | 2 | 1 |
| Len | 1 | 2 | 2 | 3 | 1 | 1 | 1 |

依序做到最後可以得到

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 4 | 2 | 3 | 4 | 2 | 1 |
| LIS | 1 | 2 | 2 | 3 | 4 | 2 | 1 |

LDS做法相同，但順序相反，做完可以得到

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num | **1** | **4** | **2** | **3** | **4** | **2** | **1** |
| LDS | 1 | 4 | 2 | 3 | 3 | 2 | 1 |

取LIS和LDS相對位置中較小值

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LIS | 1 | 2 | 2 | 3 | 4 | 2 | 1 |
| LDS | 1 | 4 | 2 | 3 | 3 | 2 | 1 |

最長序列長度 = 3\*2-1=5

方法二 :

以 1 2 5 4 5 3為範例

Len初始化為1

LIS為放入序列中的數字，每次加入新的數字會和最後一個做比較，若大於直接加入，若此數字小於LIS目前的最後一項，找到序列中大於此數字的最小值，並修正為此數字，長度會是此數字的位置

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 2 | 5 | 4 | 5 | 3 |
| Len | 1 | 1 | 1 | 1 | 1 | 1 |
| LIS | 1 |  |  |  |  |  |

因為2大於1，直接加入LIS，長度修正為2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 2 | 5 | 4 | 5 | 3 |
| Len | 1 | 2 | 1 | 1 | 1 | 1 |
| LIS | 1 | 2 |  |  |  |  |

因為5大於2，直接加入LIS，長度修正為3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 2 | 5 | 4 | 5 | 3 |
| Len | 1 | 2 | 3 | 1 | 1 | 1 |
| LIS | 1 | 2 | 5 |  |  |  |

因為4小於5，找到LIS中大於4的最小值修改，長度為3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 2 | 5 | 4 | 5 | 3 |
| Len | 1 | 2 | 3 | 3 | 1 | 1 |
| LIS | 1 | 2 | 4 |  |  |  |

因為5大於4，直接加入LIS，長度修正為4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 2 | 5 | 4 | 5 | 3 |
| Len | 1 | 2 | 3 | 3 | 4 | 1 |
| LIS | 1 | 2 | 4 | 5 |  |  |

因為3小於5，找到LIS中大於3的最小值修改，長度為3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Num | 1 | 2 | 5 | 4 | 5 | 3 |
| Len | 1 | 2 | 3 | 3 | 4 | 3 |
| LIS | 1 | 2 | 3 | 5 |  |  |

取LIS和LDS相對位置中較小值

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| LIS | 1 | 2 | 3 | 3 | 4 | 3 |
| LDS | 1 | 1 | 3 | 2 | 2 | 1 |

最長序列長度 = 3\*2-1=5

**討論：**

LIS實作有不同的方法

若使用方法一，時間複雜度O(n^2)會造成TLE

因此這題使用方法二時間複雜度O(nlogn)較適當

**程式：**

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

int num[10000];

int len\_LIS[10000];

int len\_LDS[10000];

void LIS(int n){

vector<int> v;

int size=1;

v.clear();

v.push\_back(num[0]);

for(int i=1; i<n; i++){

if(num[i]>v.back()){

v.push\_back(num[i]);

len\_LIS[i]=++size;

}

else{

int pos = lower\_bound(v.begin(), v.end(), num[i])-v.begin();

v[pos] = num[i];

len\_LIS[i] = pos+1;

}

}

}

void LDS(int n){

int size=1;

vector<int> v;

v.clear();

v.push\_back(num[n-1]);

for(int i=n-2; i>=0; i--){

if(num[i]>v.back()){

v.push\_back(num[i]);

len\_LDS[i]=++size;

}

else{

int pos = lower\_bound(v.begin(), v.end(), num[i])-v.begin();

v[pos] = num[i];

len\_LDS[i] = pos+1;

}

}

}

int cmp(int n){

int max=1;

int temp;

for(int i=0; i<n; i++){

if(len\_LIS[i]<len\_LDS[i]) temp=len\_LIS[i];

else temp=len\_LDS[i];

if(temp>max) max=temp;

}

return max;

}

int main(){

int n;

while(cin>>n){

for(int i=0; i<n; i++){

num[i]=0;

len\_LIS[i]=1;

len\_LDS[i]=1;

}

for(int i=0; i<n; i++) cin>>num[i];

LIS(n);

LDS(n);

/\* for(int i=0; i<n; i++) cout<<len\_LIS[i]<<" ";

cout<<endl;

for(int i=0; i<n; i++) cout<<len\_LDS[i]<<" ";

cout<<endl;\*/

cout<<2\*cmp(n)-1<<endl;

}

}