Longest Increasing Subsequence (LIS)

資訊科技系
林偉川

Longest Increasing Subsequence

- 最長遞增子串列定義如下:
- INPUT: 整數數列 $X=x_1,x_2,\ldots,x_n$
- OUTPUT:最長遞增子串列為此數列之部分子數列，數列具有特性為數列中之數的值比之前的數要大(each number in the sequence is larger than the previous number)
- 當此數列不唯一時，則選取其中最大子串列

2

Longest Increasing Subsequence範例

- 例如數列為{3,1,3,2}其LIS為{1,3}
- 例如數列為{5,5,18,37,4,13}其LIS為{5,18,37}
- 例如數列為{9,15,7,6,11,12,4}其LIS為{9,11,12}
- 例如數列為{9,5,2,8,7,3,1,6,4}其LIS為{2,3,6}
- 例如數列為{9,3,7,9,28,45,2,63,72,1}其LIS為{3,7,9,28,45,63,72}

3

Longest Increasing Subsequence使用的方法

- 使用長度計算(length)
- 紀錄其先前符合件之最大數(predecessor)
- 例如數列為{3,1,3,2}其LIS為{1,3}

	3	1	3	2
length	1	1	2	2
predecessor	-	-	1	1

- 例如數列為{5,5,18,37,4,13}其LIS為{5,18,37}

	5	5	18	37	4	13
length	1	1	2	3	1	2
Predecessor	-	-	5	18	-	5

4

Longest Increasing Subsequence使用的方法

- 例如數列為{9,15,7,6,11,12,4}其LIS為{9,11,12}

	9	15	7	6	11	12	4
length	1	2	1	1	2	3	1
predecessor	-	9	-	-	9	11	-

- 例如數列為{9,5,2,8,7,3,1,6,4}其LIS為{2,3,6}

	9	5	2	8	7	3	1	6	4
length	1	1	1	2	2	2	1	3	3
predecessor	-	-	-	5	5	2	-	3	3

5

Longest Increasing Subsequence使用的方法

int len[]=new int[da.length]; //使用長度計算

int pos[]=new int[da.length]; // 紀錄其先前符合件之最大數的位置

int val[]=new int[da.length]; // 紀錄其先前符合件之最大數的值

int res[]=new int[da.length]; // 要印出結果的值陣列

6

Longest Increasing Subsequence使用的方法

一開始的狀態

```
for (k=0; k<da.length; k++) {
  len[k]=1;
  pos[k]=val[k]=-1;
}
```

7

Longest Increasing Subsequence使用的方法

開始算長度及填入紀錄值

```
for (k=1; k<da.length; k++) {
    tp=0;
    for (s=0; s<k; s++) {
      if (da[k]>da[s]) {
        tp=len[s]+1;
        if (tp > len[k]) {
          len[k]=tp; pos[k]=s; val[k]=da[s];
        }
```

8

Longest Increasing Subsequence使用的方法

```
        else {
           if (val[k] != -1) {//紀錄值為其中最大者
              if (val[k] < da[s]) { pos[k]=s; val[k]=da[s]; }
           }
           else { pos[k]=s; val[k]=da[s];}
        }
      } // if (tp > len[k])
    } // if (da[k]>da[s])
    if (tp == 0) len[k]=1; //若沒有比較大的值則填1
  }
```

9

Longest Increasing Subsequence使用的方法

```
for (k= 0; k<da.length; k++) // 把表印出來
 System.out.println("da["+k+"]="+da[k]+
   " len="+len[k]+" pre="+val[k]+"pos="+pos[k]);

for (k=0; k<da.length; k++) // 找出長度最大者
   if (k==0) { max=len[k]; tp=0; }
   else if (len[k] >= max)
        if (len[k] == max) { if (da[k] > da[tp]) tp=k; }
        else { max=len[k]; tp=k; }
```

10

Longest Increasing Subsequence使用的方法

```
for (k=0; k<max; k++) {  // 將找到陣列倒著放
    res[max-1-k]=da[tp]; tp=pos[tp];  }

for (k=0; k<max; k++)  // 印出符合條件者
  if (k == 0) System.out.print(res[k]);
  else System.out.print(","+res[k]);
System.out.println("\n");
```

11