

Insertion sort and indirect sorting

08/26

002

$0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6$
 $A: 5, 3, 7, 8, 5, 0, 4$
 $\cancel{5}, \cancel{3}, \cancel{7}, 8, 5, 0, 4$
 $3, 5, \cancel{7}, 8, 5, 0, 4$

$j \quad k \quad curr \quad N$
 $\boxed{1} \quad \boxed{0} \quad \boxed{3} \quad \boxed{7}$
 $2 \quad 1 \quad 7$

$7 < 7$
 $6 < 7$
 $1 < N$

```

void insertion_sort(int A[], int N) {
    int j, k, curr;
    for (j=1; j<N; j++){
        curr = A[j];
        // insert curr (A[j]) in the
        // sorted sequence A[0...j-1]
        k = j-1;
        while ((k>=0) && (A[k]>curr)) {
            A[k+1] = A[k];
            k = k-1;
        }
        A[k+1] = curr; // - - -
    }
}

```

Instr	J	K	curr	N	descriptio
$j=1$	1			7	
$j < N$ $1 < 7$					T
$curr = A[j]$ $curr = 3$			3		
$k = j-1$ $k = 0$		0			
$k \geq 0 \ \&\& \ A[k] > curr$ $0 \geq 0 \ \&\& \ 5 > 3$ $A[k+1] = A[k]$ $A[1] = A[0]$					T
$k = k-1$ $0-1 = -1$		-1			
$k \geq 0 \ \&\& \ -1 \geq 0$					F
$A[k+1] = curr$ $A[-1+1] = 3$					
$j++$	2				
$j < N$ $2 < 7$					T
$curr = A[j]$ $= 7$					
$k = j-1$ $k = 1$		1			
$k \geq 0 \ \&\& \ A[k] > curr$ $1 \geq 0 \ \&\& \ 5 > 7$					F
$A[k+1] = curr$ $A[2] = 7$					

Insertion sort: Time Complexity & Data moves

Each row shows the array after one iteration of the outer loop for each algorithm.

'Data move' is an assignment. (Implementation will matter: deep copy or pointer)

Each row shows the array after one iteration of the outer loop for each algorithm.

Time complexity

Insert the next element in its place in the sorted sequence to the left of it.

original	5	3	7	8	5	0	4
1 st	3	5	7	8	5	0	4
2 nd	3	5	7	8	5	0	4
3 rd	3	5	7	8	5	0	4
4 th	3	5	5	7	8	0	4
5 th	0	3	5	5	7	8	4
6 th	0	3	4	5	5	7	8

Gray cells are visited by the iterations of the inner loop => they are proportional with the time complexity => $\sim N^2/2$ (worst case)

Data moves

Insert the next element in its place in the sorted sequence to the left of it.

5	3	7	8	5	0	4
3	5	7	8	5	0	4
3	5	7	8	5	0	4
3	5	7	8	5	0	4
3	5	5	7	8	0	4
0	3	5	5	7	8	4
0	3	4	5	5	7	8

Each red number: 2 moves.

Each blue number: 1 move.

Best: $2(N-1)$ Worst: $2(N-1)+N(N-1)/2$

Average: $2N+N(N-1)/4$

Indirect Sorting



Handwritten notes at the top: $Data[i]$, $Data[X[0]]$, cur , 201

X	Data	Ordered	j	X[j]	Data[X[j]]
0 1	709	115	0	$X[0] \rightarrow 1$	$Data[1] \rightarrow 115$
1 3	115	201	1	$X[1] \rightarrow 3$	$Data[3] \rightarrow 201$
2 5	616	427	2		
3 4	201	505	3		
4 2	505	616	4		
5 0	427	709	5		
6 6	934	934	6	$X[6] \rightarrow 6$	$Data[6] \rightarrow 934$

Handwritten annotations on the table:

- Arrows from the 'X' column to the 'Data' column: $X[1] \rightarrow 1$, $X[2] \rightarrow 2$, $X[3] \rightarrow 3$, $X[4] \rightarrow 4$, $X[5] \rightarrow 5$.
- Arrows from the 'Data' column to the 'Ordered' column: $Data[X[0]] \rightarrow 115$, $Data[X[1]] \rightarrow 201$, $Data[X[2]] \rightarrow 427$, $Data[X[3]] \rightarrow 505$, $Data[X[4]] \rightarrow 616$, $Data[X[5]] \rightarrow 709$, $Data[X[6]] \rightarrow 934$.
- Text: "The i-th element in sorted order is given by $Data[Idxs[i]]$ ".

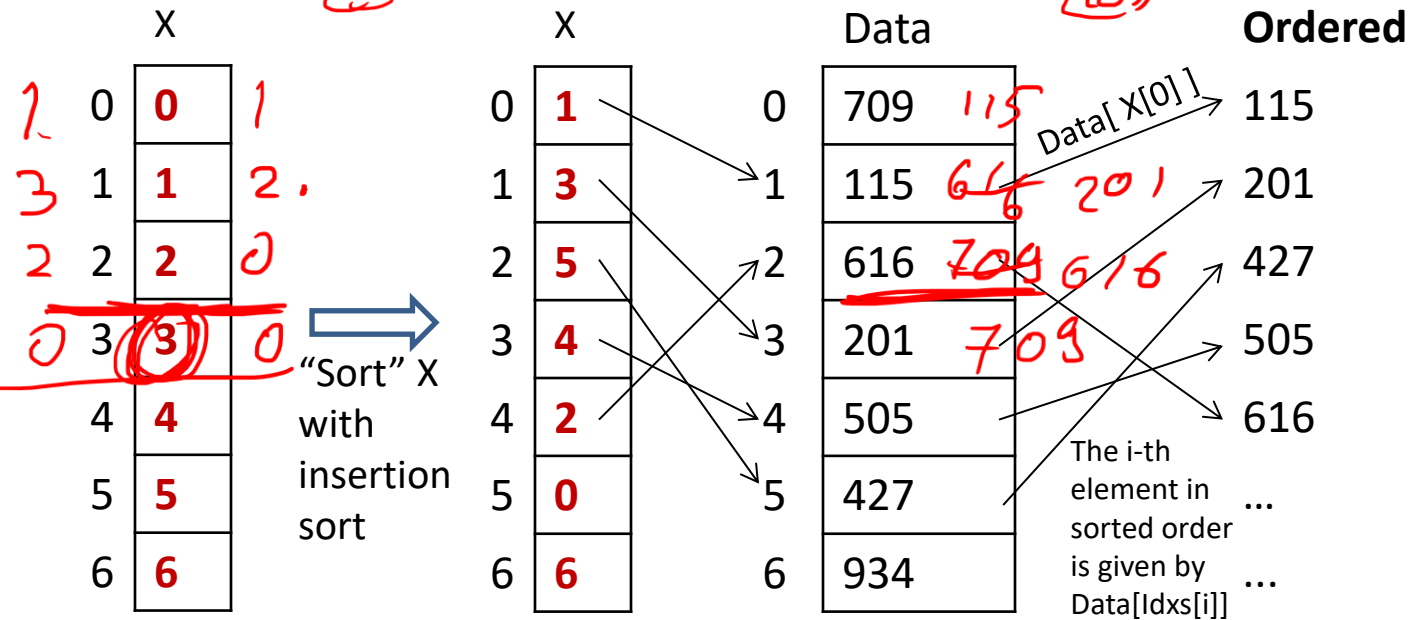
```
for (j=0; j<7; j++){
    printf("%d\n", Data[ X[j] ] );
}
```

Handwritten red annotations: $Data[3]$ is written below the $X[j]$ in the printf statement.

Indirect Sorting

Data[x]? Data[x[k]] curr

C
[3]



```
void insertion_sort(int A[],int N){
    int j,k,curr;
    for (j=1; j<N; j++){
        curr = A[j];
        k = j-1;
        while ((k>=0) && (A[k] > curr)){
            A[k+1] = A[k];
            k = k-1;
        }
        A[k+1] = curr;
    }
}
```

C
i = X[i]

1. Create the identity array, X, with indexes 0 to N-1
2. Adapt insertion sort to rearrange the elements of X
 1. What do you copy?
 2. What do you compare?
3. Return X
 1. Language specific issues (C/Java)

Access Data through X: e.g. Data[X[j]]