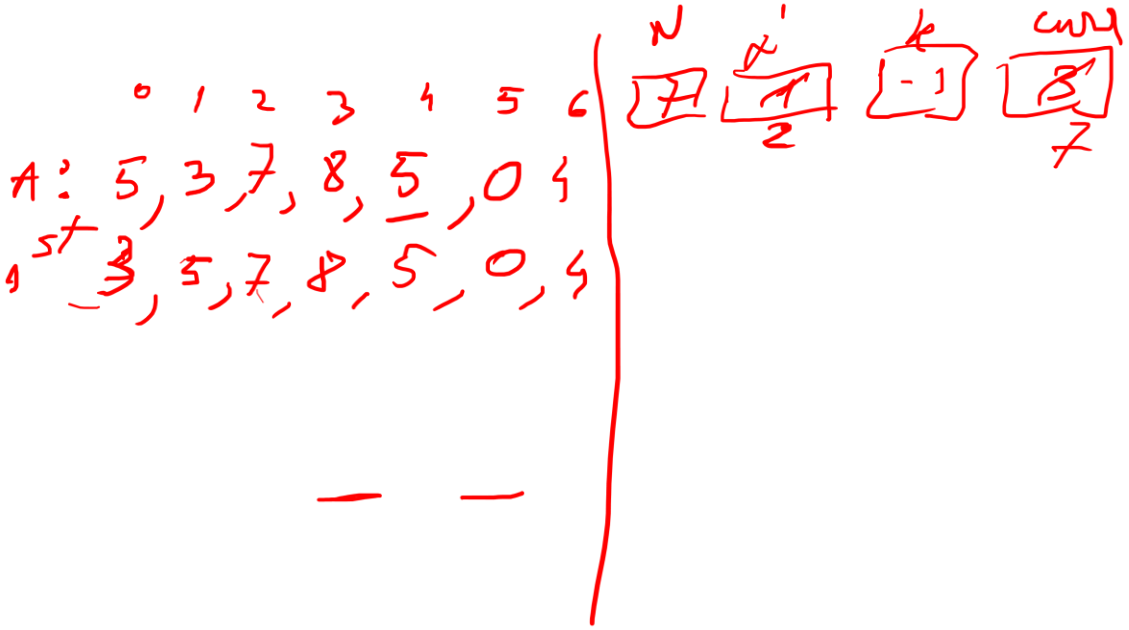
Insertion sort and indirect sorting

08/26
001



Instr	j	K	curr	N	descriptio
j := 1	1				
j < N					
curr = A[j]			3		
k = j - 1		0			
k >= 0 && A[k] > curr					True
A[k+1] = A[k]					A modified
A[1] = 5					
k--		-1			
k >= 0 && A[k] > curr					False
A[k+1] = curr					
A[-1+1] = 3					
A[0] = 3					
j++	2				
j < N					T
2 < 7					
curr = A[j]		1	7		

```
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;
}
```

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 it's place in the sorted sequence to the left of it.

original	5	3	7	8	5	0	4
1st	**3**	**5**	7	8	5	0	4
2nd	3	5	**7**	8	5	0	4
3rd	3	5	7	**8**	5	0	4
4th	3	5	**5**	7	8	0	4
5th	**0**	3	5	5	7	8	4
6th	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 => ~ $N^2/2$ (worst case)

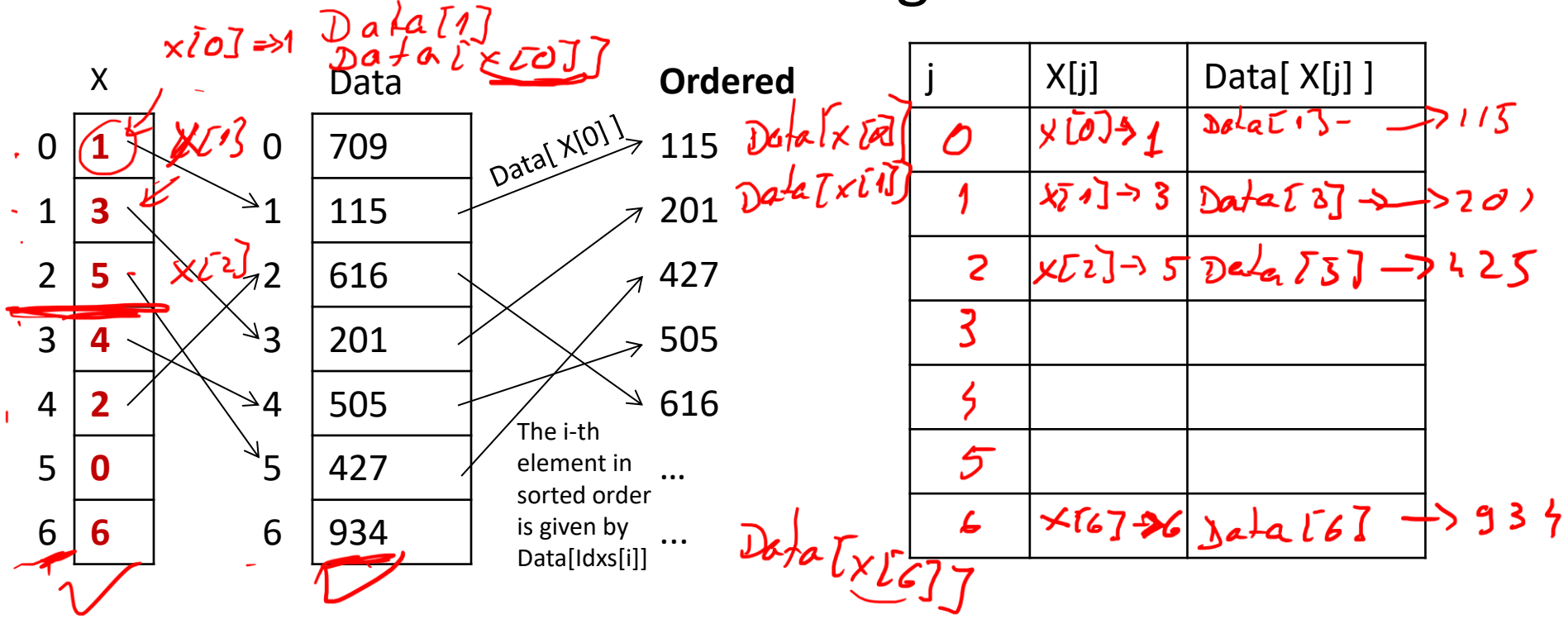
Data moves

Insert the next element in it's 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

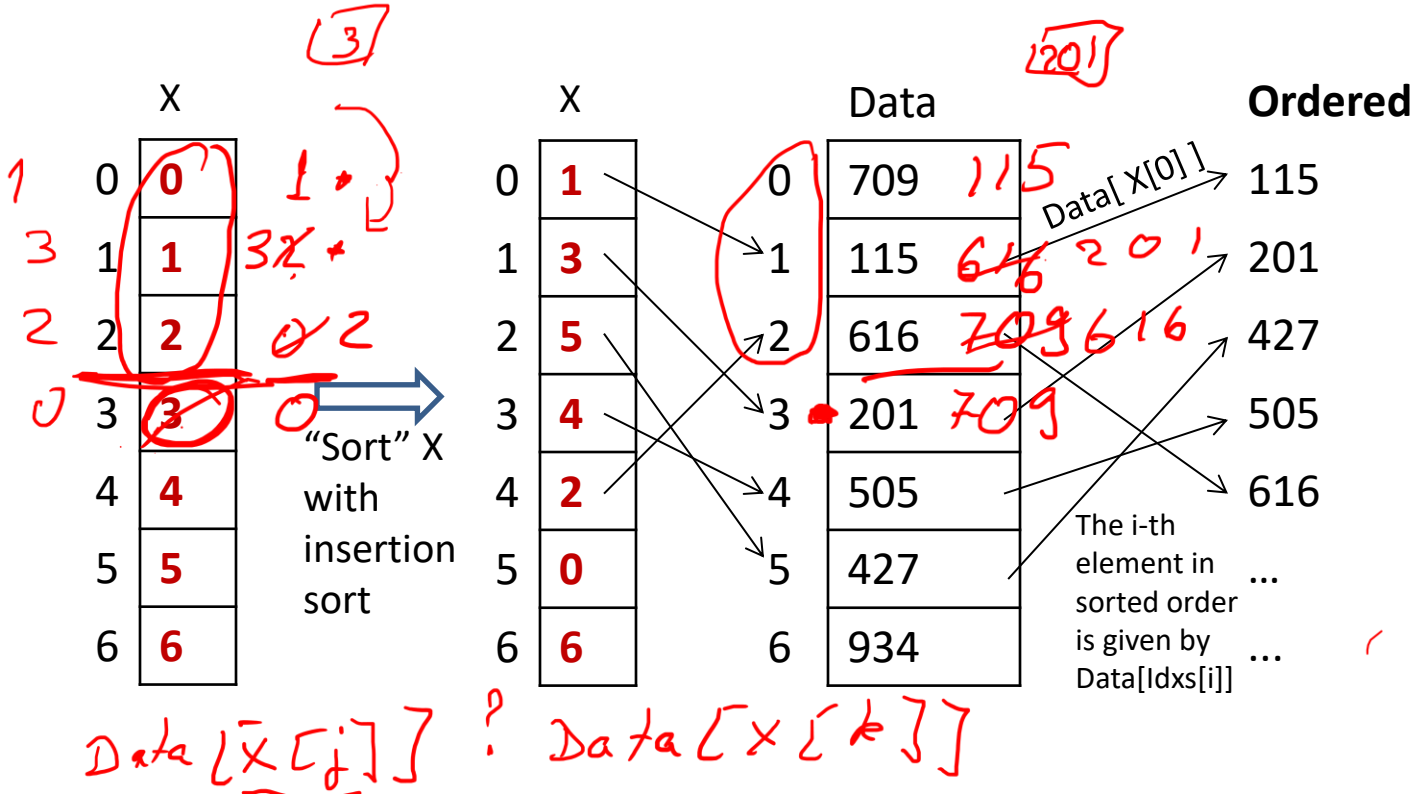
Indirect Sorting



j	X[j]	Data[X[j]]
0		
1		
2		
3		
4		
5		
6		

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

Indirect Sorting




```
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;
}
```

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]]