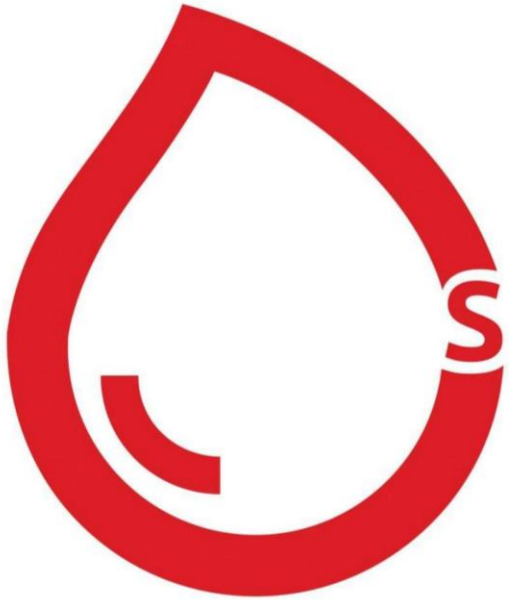


ورشة الخوارزميات

إعداد فريق الكريات الحمراء

محاور هذه الجلسة:

- Complexity
- Sorting algorithms
- Searching algorithms

Complexity

- Choose correct answer.
- Complexity of algorithm.
- Complexity of piece of code.

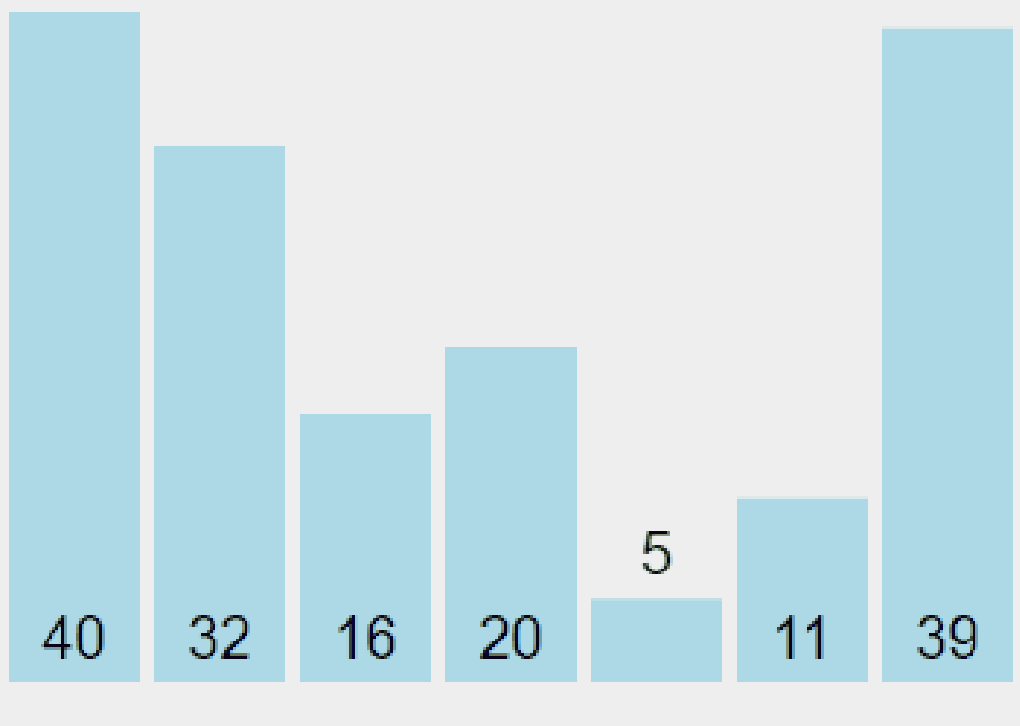
Sorting algorithms

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort

Bubble sort

- هي خوارزمية فرز تقوم على أساس تكرار تبديل الأزواج الغير مرتبة حتى يتم ترتيب عناصر المصفوفة
- تجري الخوارزمية ضمن phases في كل phase نقوم بعملية bubble up لأكبر عنصر في الجزء الغير مرتب من العناصر إلى نهاية هذا الجزء.

Bubble sort

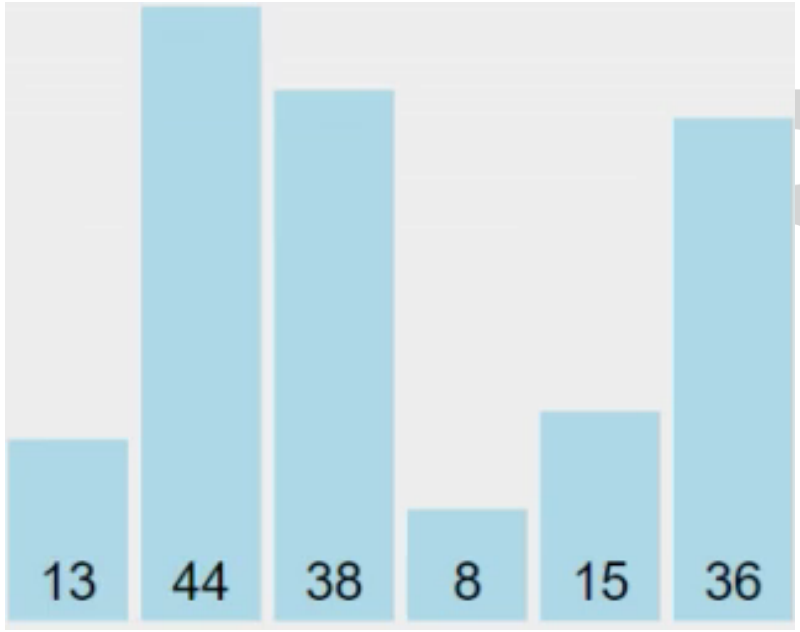


1) for i = 1 to A.length-1
2) for j = 1 to A.length-i-1
3) if(A[j] > A[j+1])
4) swap(A[j], A[j+1])

$$O(n^2)$$

Selection sort

- تقوم هذه الخوارزمية من خلال تكرار البحث عن أصغر عنصر في الجزء الغير مرتب من السلسلة ووضعه في مكانه المناسب حتى يتم ترتيب هذه السلسلة بأكملها..



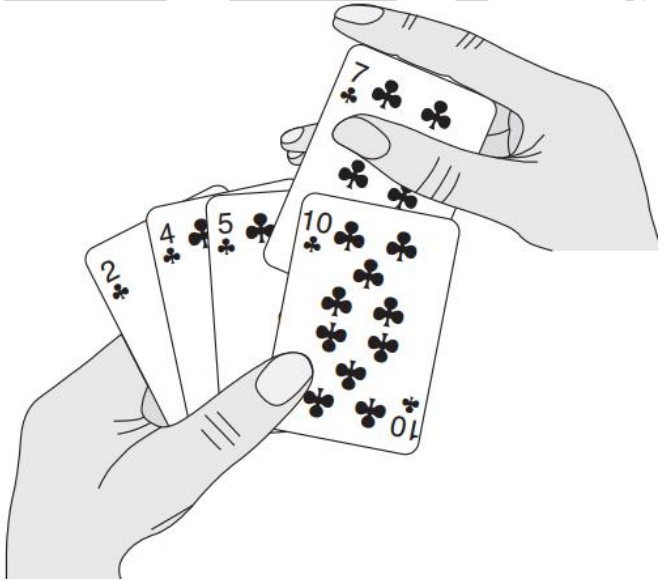

```
1.  For i=1 to n-1
2.    min=i
3.    for j=i+1 to n
4.          if (a[j]<a[min])
5.              min=j
6.    if (min != i)
7.          swap(a[i],a[min])
```

$O(n^2)$

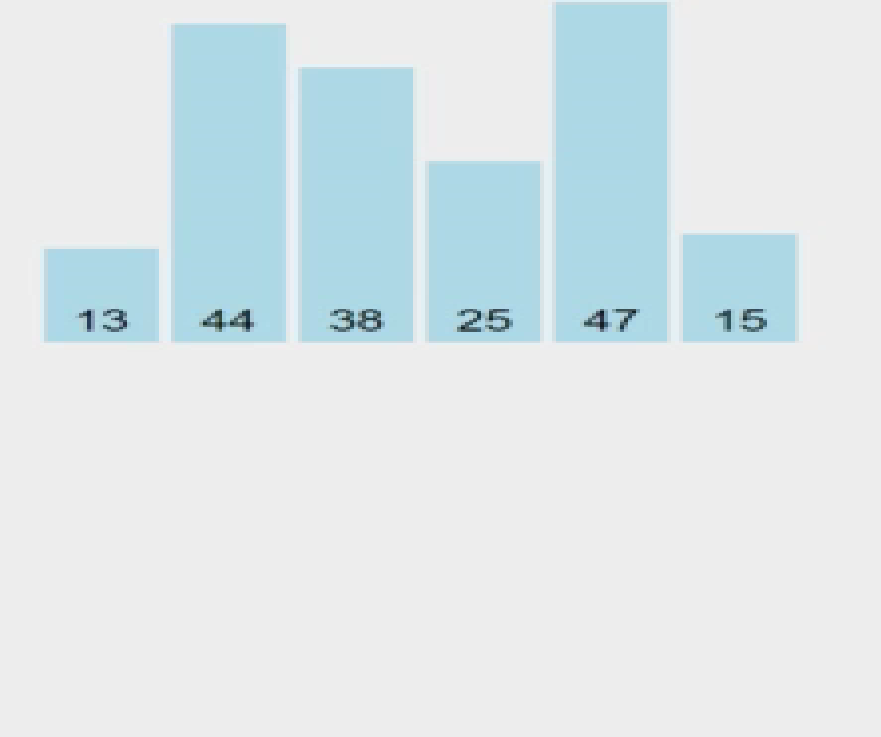
Insertion sort

- تقوم هذه الخوارزمية بافتراض جزء مرتب في سلسلة من العناصر (عنصر وحيد في أسوأ الحالات)

و القيام بعملية insert لكل العناصر الأخرى الى هذا الجزء المرتب عنصرا تلو الاخر...



Insertion sort




```
1.  for i=2 to n
2.      key = a[i]
3.      j=i-1
4.      while (j>0 and a[j]>key)
5.          a[j+1]=a[j]
6.          j=j-1
7.      a[j+1]=key
```

$O(n^2)$

Merge

4	6	8	13

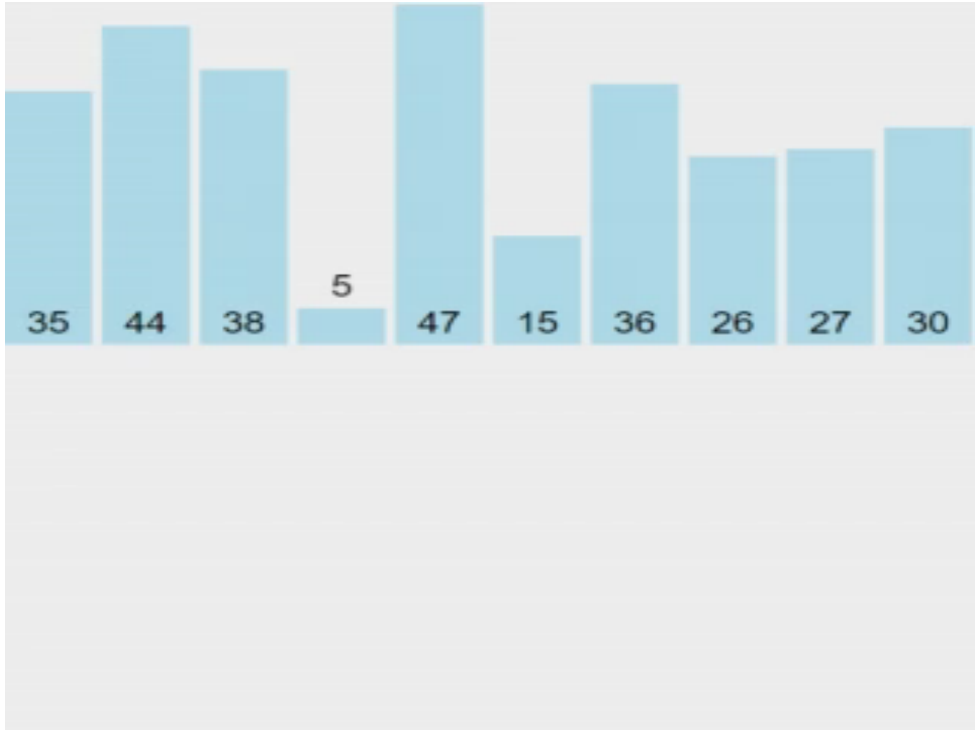
1	7	11	14

- بفرض لدينا مصفوفتين مرتبتين:

نريد دمجهما في مصفوفة جديدة مرتبة.

1	4	6	7	8	11	13	14

Merge sort




```
Merge-Sort(A,Left,Right)
1.    If Left < Right
2.        mid=[(Left + Right)/2]
3.        Merge-Sort(A,Left,mid)
4.        Merge-Sort(A,mid+1,Right)
5.        Merge(A,Left,mid,Right)
```

Quick sort

QuickSort(A,p, r)

1. if p < r
2. q=Partition(A,p, r)
3. QuickSort(A,p, q-1)
4. QuickSort(A,q+1, r)

Best-Case:

O(n.log(n))

Worst-Case:

O(n^2)

Searching algorithms

- Linear Search
- Binary Search

Linear Search

- تقوم بهذه الخوارزمية بمسح العناصر جميعها حتى نجد العنصر المطلوب أو لا نجده...
- تعقيد هذه الخورازمية : $O(n)$

Binary search

- يشترط لتطبيق هذه الخوارزمية ان تكون العناصر مرتبة..
- بفرض لدي مجموعة عناصر مرتبة

4	7	13	20	21	26	29	37	39

Binary search

```
binarySearch(arr, item, beg, end)
if beg<=end
midIndex = (beg + end) / 2
if item == arr[midIndex]
return midIndex
else if item < arr[midIndex]
return binarySearch(arr, item, midIndex + 1, end)
else return binarySearch(arr, item, beg, midIndex - 1)
return -1
```