

① Bubble sort, Insertion sort, Merge sort, Quick sort, heap sort

A) random array: (Average case)

(أي جملة مختلطة من الأرقام)

Bubble sort: $\Theta(n^2)$, Insertion sort: $\Theta(n^2)$, Merge sort: $\Theta(n \log n)$

Quick sort: $\Theta(n \log n)$, heap sort: $\Theta(n \log n)$

Quick sort: $\Theta(n \log n)$, heap sort: $\Theta(n \log n)$

(أفضل الحالات) (أسوأ الحالات)

Quick sort > heap sort > Merge sort > Insertion sort > Bubble sort

B) nearly sorted: (Best case) \rightarrow Insertion sort

Bubble sort: $\mathcal{O}(n)$, Insertion sort: $\mathcal{O}(n)$, Merge sort: $\mathcal{O}(n \log n)$

Quick sort: $\mathcal{O}(n \log n)$, heap sort: $\mathcal{O}(n \log n)$

why?

Bubble sort, Insertion sort، هي أسرع من mergesort في

حالات أفضل، وفي حالات أسوأ، هي أسرع من mergesort.

Insertion sort، هي أسرع من mergesort في الحالات الأسوأ.

Bubble sort، هو أسرع من mergesort في الحالات الأسوأ.

C)

```
void BubbleSort (list<int> array) {
    len = array.Count();
    bool swapped = false;
    while (swapped == false) {
        for (int i = 1; i < len - 1; i++) {
            if (array[i - 1] > array[i]) {
                swap (array[i - 1], array[i]);
                swapped = true;
            }
        }
        len = len - 1;
    }
}
```

②

A) Quick Sort {0, 10, 10, 5, 3, 5, 10, 3, 9, 5} Pivot = P

P	0	10	10	5	3	5	10	3	9	5
	0	10	10	5	3	5	10	3	9	5

P	0	10	10	5	5	5	10	3	9	3
	0	10	10	5	5	5	10	3	9	3

→ Quick sort

0	3	10	5	5	5	10	3	9	70
0	3	10	5	5	5	10	3	9	70

P

0	3	10	5	5	5	10	3	9	10
---	---	----	---	---	---	----	---	---	----

P

0	3	10	5	5	10	10	3	9	5
---	---	----	---	---	----	----	---	---	---

P

0	3	3	5	5	10	10	10	9	5
---	---	---	---	---	----	----	----	---	---

P Quick Sort

0	3	3	5	5	10	10	10	9	5
---	---	---	---	---	----	----	----	---	---

P

0	3	3	5	5	10	10	10	9	5
---	---	---	---	---	----	----	----	---	---

P

0	3	3	5	5	10	5	10	9	10
---	---	---	---	---	----	---	----	---	----

P Quick sort

0	3	3	5	5	9	5	10	10	10
---	---	---	---	---	---	---	----	----	----

P

0	3	3	5	5	9	5	10	10	10
---	---	---	---	---	---	---	----	----	----

P Quick Sort

0	3	3	5	5	5	9	10	10	10
---	---	---	---	---	---	---	----	----	----

P

0	3	3	5	5	5	9	10	10	10
---	---	---	---	---	---	---	----	----	----

P

0	3	3	5	5	5	5	9	10	10
---	---	---	---	---	---	---	---	----	----

P

0	3	3	5	(5)	5	9	10	10	10
Quick Sort									
0	3	3	5	5	5	9	10	(10)	10
P									
0	3	3	5	5	5	9	10	10	(10)

P

0	3	3	5	5	5	9	10	(10)	10
P									
0	3	3	5	5	5	9	10	(10)	10

✓ Done sorting

B)

اگر آنرا از اصل مرتب شده باشند، مانند پایه ای کارهای داشته باشند، بینیم زمانی سریع
می شود. برای آنرا مسکوچین مثل می خواهیم اعدادی کارهای (مانند 3 و 5 و 10) را بین
آن بینیم زمانی سریع می شود.

برای حل این مشکل بینیم که در بین این کارهای بین زمانی سریع می شود، می توانیم از این
ساختار استفاده کنیم که از اینها آنهاست که سریع می شوند: سمت میانی حرکت می کند کارهای
که در اینجا مانند 3 و 5 و 10 دهند. مانند زمانی که در میان داشته باشند، می توانند ترتیب
کارهای خود را تغییر دهند. order استیاهی کارهای فرسته باشند و اینجا با $swap$ می شوند.
و در این کارهای $swap$ که Pivot که استیاهی کارهای فرسته باشند، می توانند ترتیب
که می توانند که ساخته شده باشند، الگوریتم متوقف می شود ساخته شده باشند.

C)

ب) رہائش کر دادہ میں اگر داریم کہ اس کا سائز n ہے تو اس کو Counting Sort کہا جاتا ہے۔

ایسا جسکے $O(n+k)$ ہے اسے بھی Counting Sort کہا جاتا ہے۔

ایسا جسکے range of K کو باہم میں لے لیا جائے تو اسے Bucket Sort کہا جاتا ہے۔

③

A) {13000206, 13000509, 13561130, 13670816, 13980501, 13991229, 13460824, 13780524, 13890524, 13330331}

Step 1:

13000206	13000509	13561130	13670816	13980501
----------	----------	----------	----------	----------

13991229	13460824	13780524	13890524	13330331
----------	----------	----------	----------	----------

Step 2:

13000206	13000509	13561130	13670816	13980501
----------	----------	----------	----------	----------

13991229	13460824	13780524	13890524	13330331
----------	----------	----------	----------	----------

Step 3:

13000206	13000509	13561130	13670816	13980501
----------	----------	----------	----------	----------

13991229	13460824	13780524	13890524	13330331
----------	----------	----------	----------	----------

13000206

13000509

13561130

13670816

13980501

13991229

13460824

13780524

13890524

13330331

13000206	13000509
----------	----------

13561130

13670816	13980501
----------	----------

13460824	13991229
----------	----------

13780524

13330331	13890524
----------	----------

13000206	13000509
----------	----------

13561130	13670816	13980501
----------	----------	----------

13460824	13991229
----------	----------

13330331	13780524	13890524
----------	----------	----------

13000206	13000509	13561130	13670816	13980501
----------	----------	----------	----------	----------

13330331	13460824	13780524	13890524	13991229
----------	----------	----------	----------	----------

13000206, 13000509, 13330331, 13460824, 13561130, 13670816,
13780524, 13890524, 13980501, 13991229

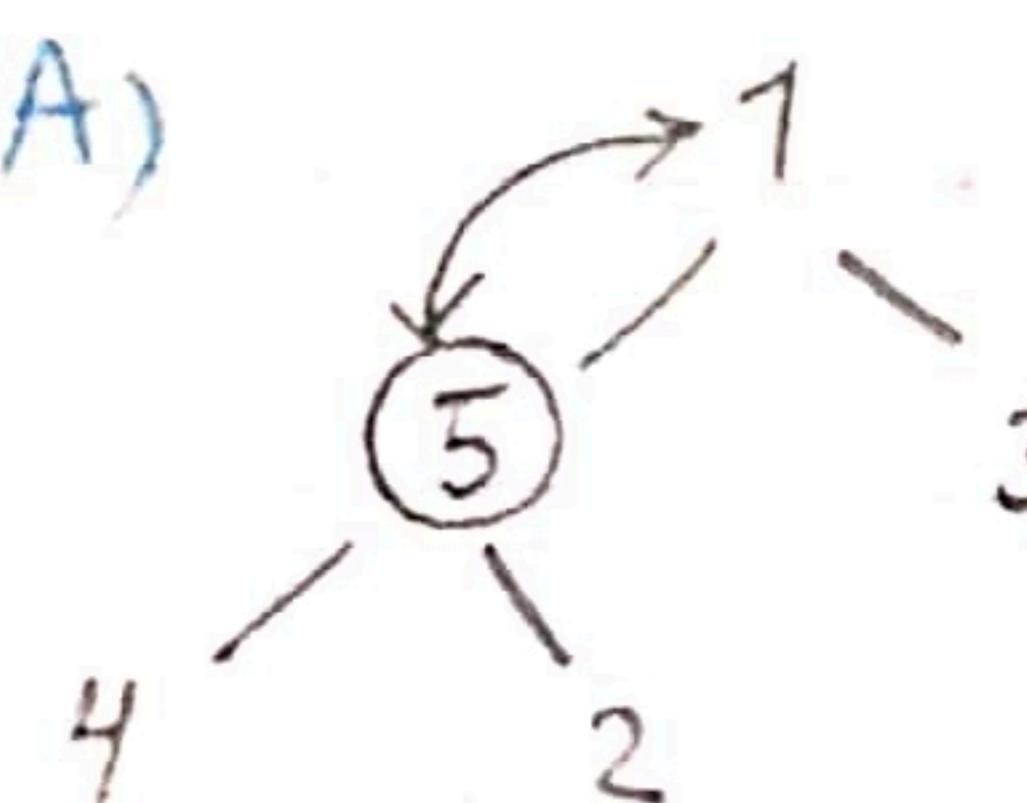
B)

sort بالطريق التقليدي حج: Radix sort

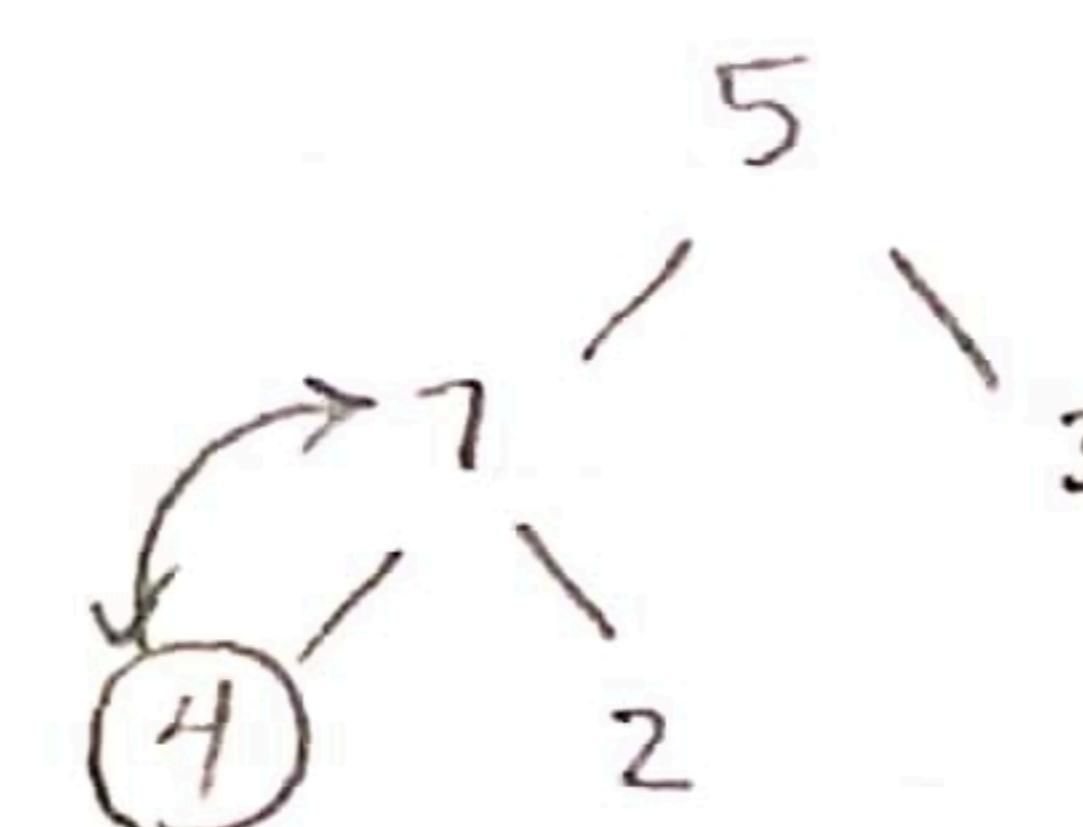
الخطوات المتبعة

④

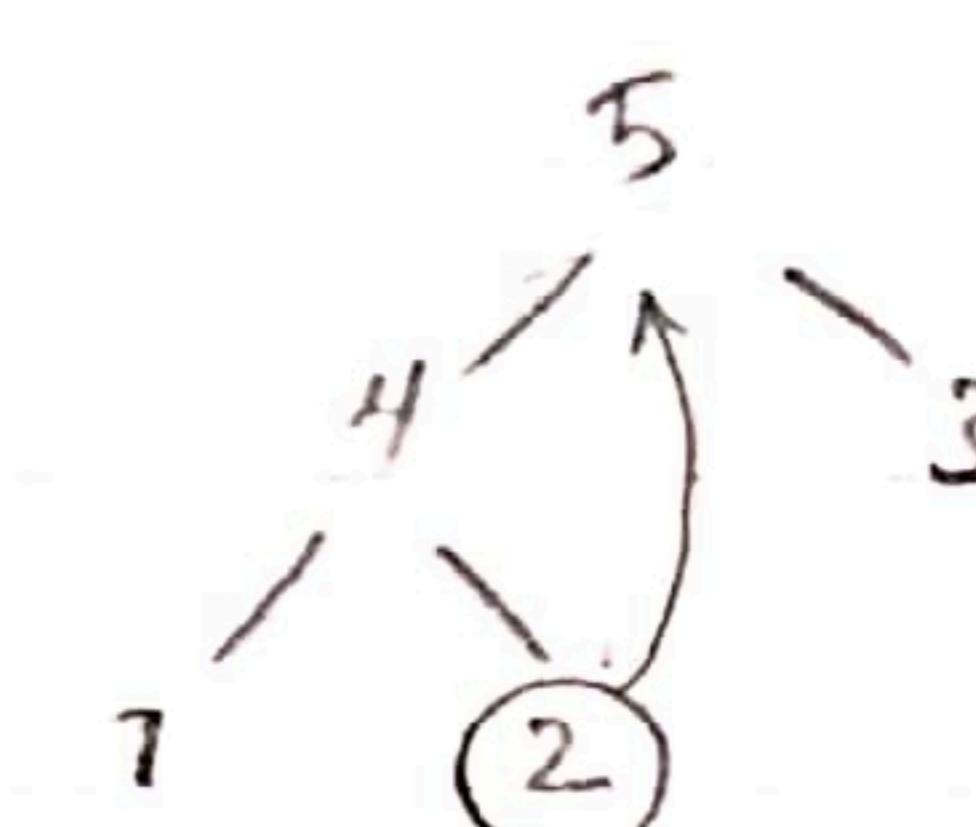
A)



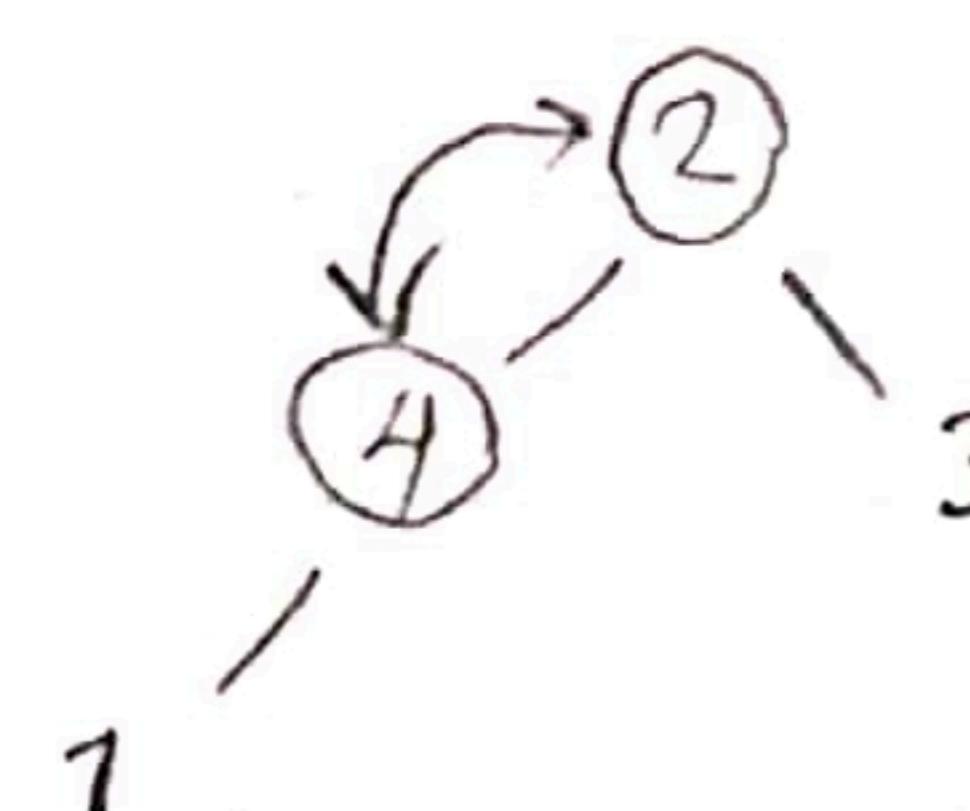
[x, x, x, x, x]



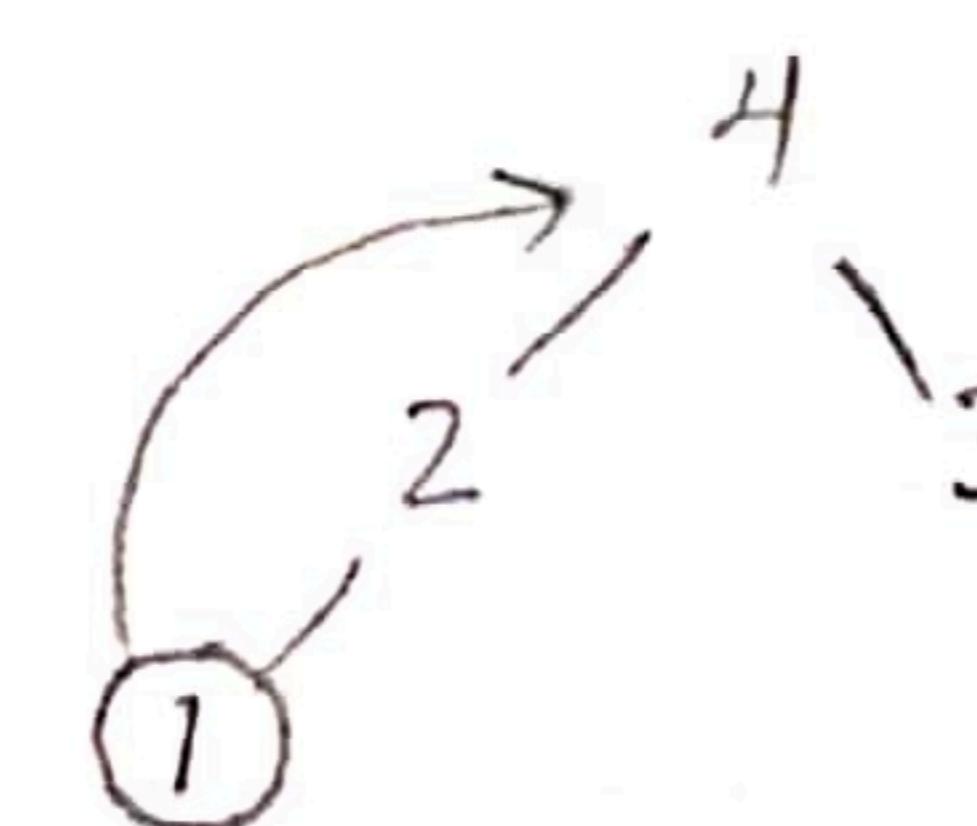
[x, x, x, x, x]



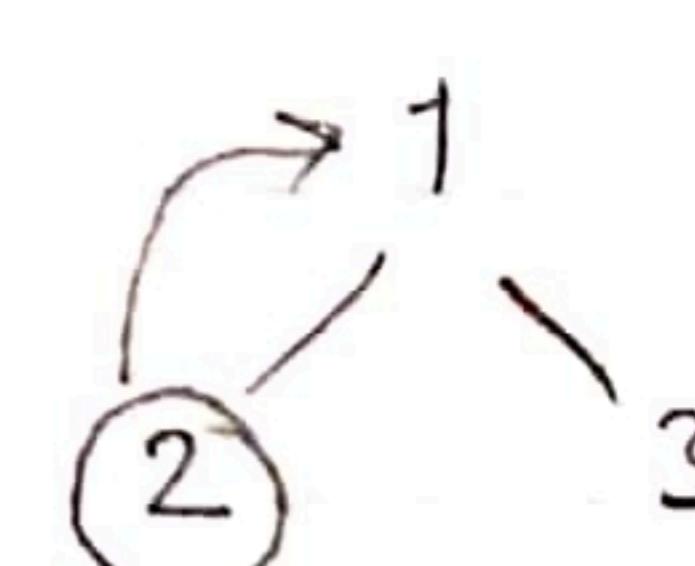
[x, x, x, x, x]



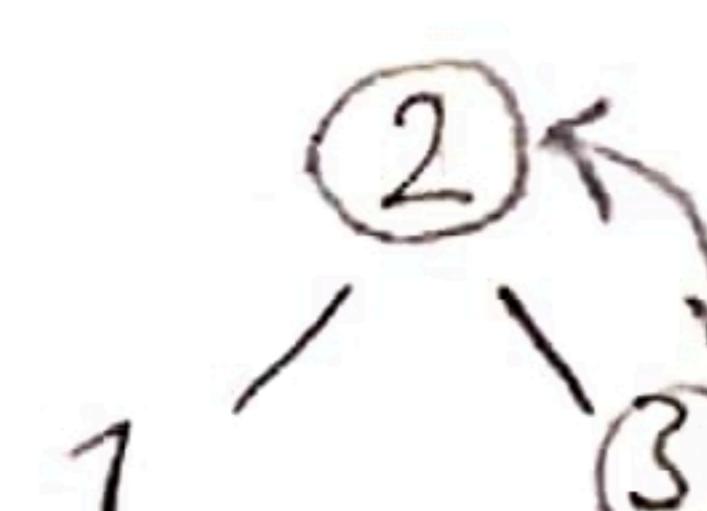
[x, x, x, x, 5]



[x, x, x, x, 5]



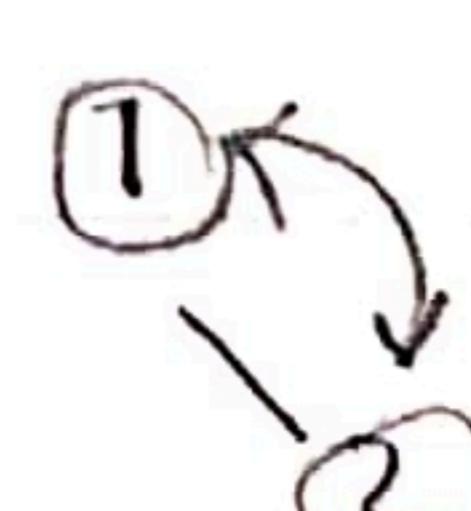
[x, x, x, 4, 5]



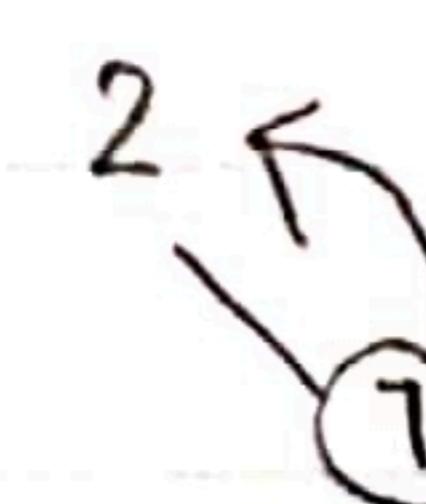
[x, x, x, 4, 5]



[x, x, x, 4, 5]



[x, x, 3, 4, 5]



1

[x, x, 3, 4, 5]

[x, 2, 3, 4, 5]

[1, 2, 3, 4, 5]

B)

• Um zu unterscheiden: $O(n \log n)$ für $\max_{1 \leq i \leq n} a_i$ sortiert und
durch $\max_{1 \leq i \leq n} a_i$ teilt, überprüft, ob $a_i = k$