ساختمان داده ها

دانشگاه صنعتی نوشیروانی بابل

دکتر حسام عمران پور

طراحان اسلاید:
زهرا ریحانیان و دانیال علیزاده

حل تمرین :
علی باقری

لینک کانال تلگرام اطلاع رسانی و حل تمرین:
t.me/ds_nit_4011

فصل ۱۱
الگوریتم های مرتب سازی

الگوریتم های مرتب سازی را بر اساس چهار ویژگی دسته بندی می کنند :

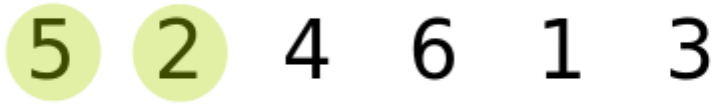
- داخلی / خارجی
- مقایسه ای / غیر مقایسه ای
- درجا / برون جا
- پایدار / ناپایدار

مرتب سازی درجی

مرتب سازی درجی

```
INSERTION-SORT ( L : ARRAY; n : integer )
{
 // L is an array of size n
   for i = 2 to n do
   {
      Current = L[i];
       j = i – 1;
      while j 3 1 and L[i] > Current do
      {
          L[j + 1] = L[j];
          j-- ;
       }
       L[j + 1] = Current;
   }
}
```

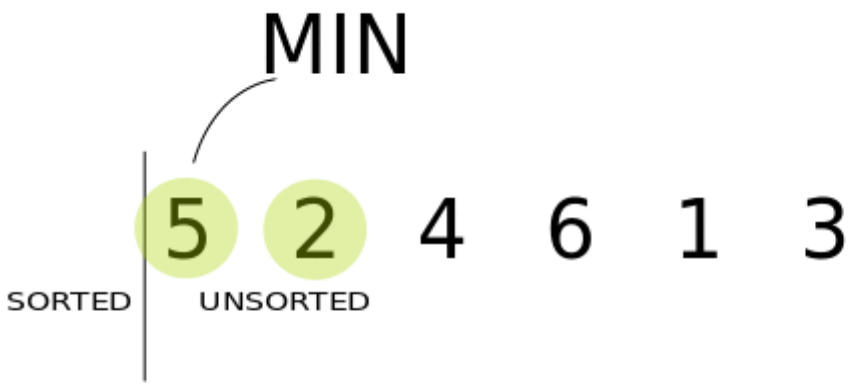
مرتب سازی حبابی



مرتب سازی حبابی

```
BUBBLE-SORT( L: ARRAY; n: integer )
{
 // L is an array of size n
    for i = 1 to n – 1 do
   {
       Swapped = FALSE;
       for j = 1 to n – i do
      {
            if L[j] > L[j+1] then
           {
                 Swap(L[j], L[j+1]);
                Swapped = TRUE;
           }
       }
      if Swapped == FALSE then
          return;
    }
}
```

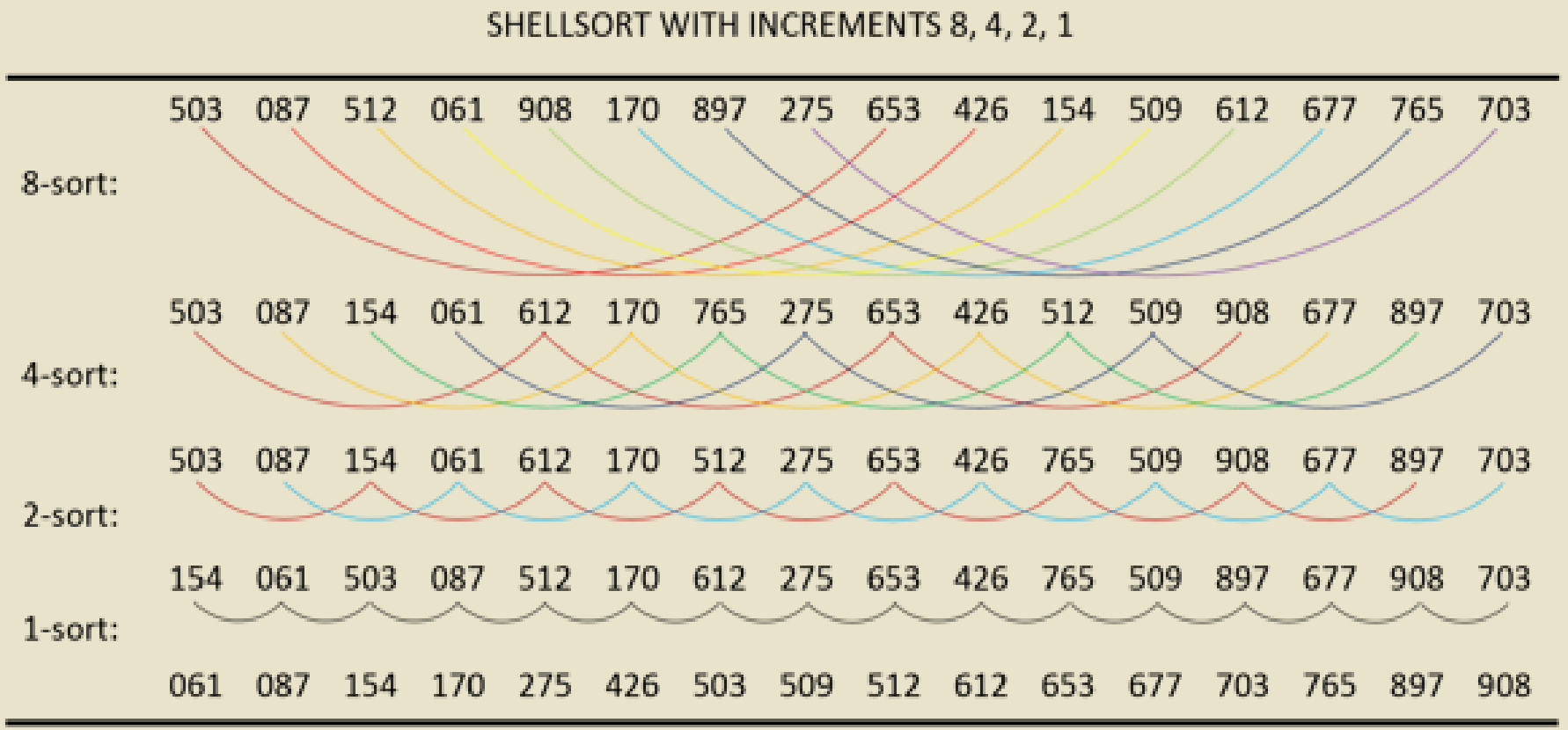
مرتب سازی انتخابی



مرتب سازی انتخابی

```
SELECTION-SORT( L: ARRAY; n: integer )
{
 // L is an array of size n
     for i = 0 to n – 2 do
     {
          maxIndex = 1;
          for j = 2 to n – i do
                if L[j] > L[maxIndex] then
          maxIndex = j;
          if n – i ≠ maxIndex then
          Swap(L[n-i], L[maxIndex]);
      }
}
```

مرتب سازی صدفی



مرتب سازی صدفی

```
SHELL-SORT( L: ARRAY; n: integer )
{
 // L is an array of size n
 var passes, gap, Current, D : integer;
 passes = ⌊log2 n⌋ ;
 while passes ≥ 1 do
 {
      gap = 2^passes;
      D = gap – 1;
      // Sort each sublist using insertion sort
      for i = gap to n do
      {
          Current = L[i];
          j = i – D;
          while j ≥ 1 and L[j] > Current do
          {
                L[j + D] = L[j];
                j = j - D;
          }
          L[j + D] = Current;
      }
      passes--;
 }
}
```

مرتب سازی هرمی

```
HEAP-SORT( A: ARRAY; n: integer )
{
    // Heapify
    for i = n / 2 downto 1 do
        SIFT-DOWN(A, i);
    // Delete Root for n times
    for i = n downto 1 do
    {
        Swap(A[1], A[i]);
        SIFT-DOWN(A, 1);
    }
}
```

مرتب سازی ادغامی

6 5 3 1 8 7 2 4

الگوریتم ادغام

```
MERGE(L: ARRAY; start1, end1, start2, end2: integer){
    // Merges L1[start1..end1] and L2[start2..end2]
    // Result is an array with size of size1+size2
    var size1, size2 : integer;
    size1 = end1 – start1 + 1;
    size2 = end2 – start2 + 1;
    // Size of temp array is (size1 + size2)
    var temp : ARRAY;
    // i1 = index on L1, i2 = index on L2;
    // it = index on temp
    var i1, i2, it: integer;
    i1 = start1;
    i2 = start2;
    it = 1;
    while i1 ≤ end1 and i2 ≤ end2 do{
        if L[i1] ≤ L[i2] then {
            temp[it] = L[i1];
            i1++ ;
        }
        else {
            temp[it] = L[i2];
            i2++ ;
        }
        it++ ;
    }
    // Copy the remaining elements
    while i1 ≤ end1 do {
        temp[it] = L[i1];
        i1++;
        it++;
    }
    while i2 ≤ end2 do {
        temp[it] = L[i2];
        i2++;
        it++;
    }
    // Copy result into the main list (L)
    for i = 1 to size1 + size2 do
        L[start1 + i - 1] = temp[i];
}
```

مرتب سازی ادغامی

```
MERGE-SORT(L: ARRAY; first, last: integer)
{
      // L is an array of size n, n=last–first+1
      if first ≥ last then
            return;
      middle = (first + last) / 2;
      MERGE-SORT(L, first, middle);
      MERGE-SORT(L, middle + 1, last);
      MERGE(L, first, middle, middle + 1, last);
}
```

$T(n) = 2\,T(\frac{n}{2}) + O(n)$ → $O(n \log n)$

مرتب سازی سریع

6 5 3 1 8 7 2 4

مرتب سازی سریع

۱. اگر تعداد عناصر صفر یا یک است ، بازگشت کن.

۲. یک عنصر را به روشی (که بعدا تشریح می شود) انتخاب کن و آن عنصر را به عنوان محور در نظر بگیر.

۳. مجموعه ی عناصر را به دو لیست L و R افراز (بخش بندی) کن به طوری که

$$\forall x \in L, x \leq pivot$$
$$\forall x \in R, x > pivot$$

۴. الگوریتم مرتب سازی سریع را بر روی لیست های L و R به طور مجزا اجرا کن.

۵. نتیجه را به صورت

L	pivot	R

لیست بازگردان.

مرتب سازی سریع

روش های رایج تعیین محور

- انتخاب اولین عنصر لیست به عنوان محور
- انتخاب آخرین عنصر لیست به عنوان محور
- انتخاب عنصر وسط لیست به عنوان محور
- بررسی کردن عنصر اول، وسط و آخر لیست و انتخاب عنصر میانه به عنوان محور
- انتخاب تصادفی یک عنصر از لیست به عنوان محور
- و غیره.

مرتب سازی سریع

```
QUICK-SORT(L: ARRAY; first, last: integer)
{
    // L is an array of size n, n=last–fist+1
    if last – first + 1 < cutoff then
        INSERTION-SORT(L, last – first + 1);
    else
    {
        pivotIndex = PARTITION(L, first, last);
        QUICK-SORT(L, first, pivotIndex– 1);
        QUICK-SORT(L, pivotIndex + 1, last);
    }
}
```

```
PARTITION (L: ARRAY; first, last: integer)
{
      middle = (first + last) / 2;
      // MEDIAN function returns the index of
      // the median value by comparing L[first],
      // L[last] and L[middle]
      pivotIndex = MEDIAN(L, first, middle, last);
      pivot = L[pivotIndex];
      // Place pivot at the last position of list
      Swap(L[pivotIndex], L[last]);
      // Begin partitioning
      i = first;
      j = last – 1;
      while TRUE do
      {
            while i < last and L[i] ≤ pivot do
                  i++ ;
            while j > 0 and L[j] > pivot do
                  j-- ;
            if i < j then
                  Swap(L[i], L[j]);
            else
                  break;
      }
      // End of partitioning
      // Put pivot between L and R sublists
      Swap(L[last], L[i]);
     // Return the position of pivot
     return i;
}
```

بهترین حالت pivot در وسط

$$T(n) = 2\,T\left(\frac{n}{2}\right) + n = \theta(n \log n)$$

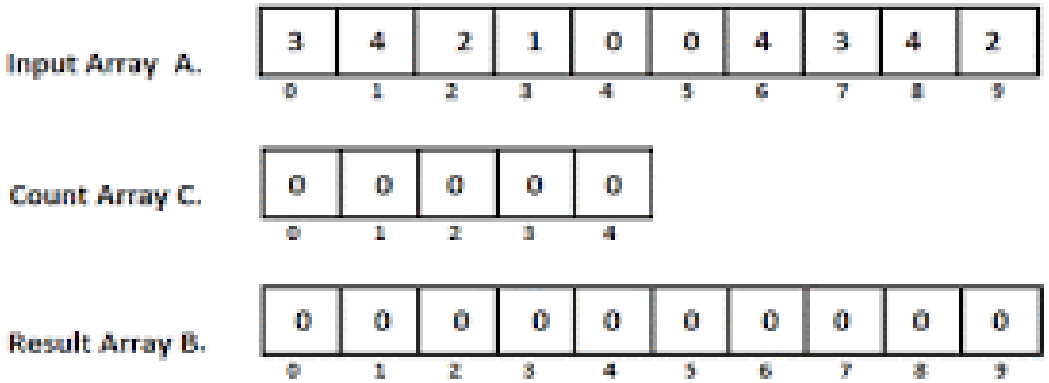
بدترین حالت pivot در اول یا
آخر لیست قرار گرفته باشد

$$T(n) = 2\,T(n-1) + n = \theta(n^2)$$

مرتب سازی شمارشی

Counting Sort...

Counting Sort... N=10 , K=5



مرتب سازی شمارشی

Counting Sort... N=10 , K=5

Count Array C.

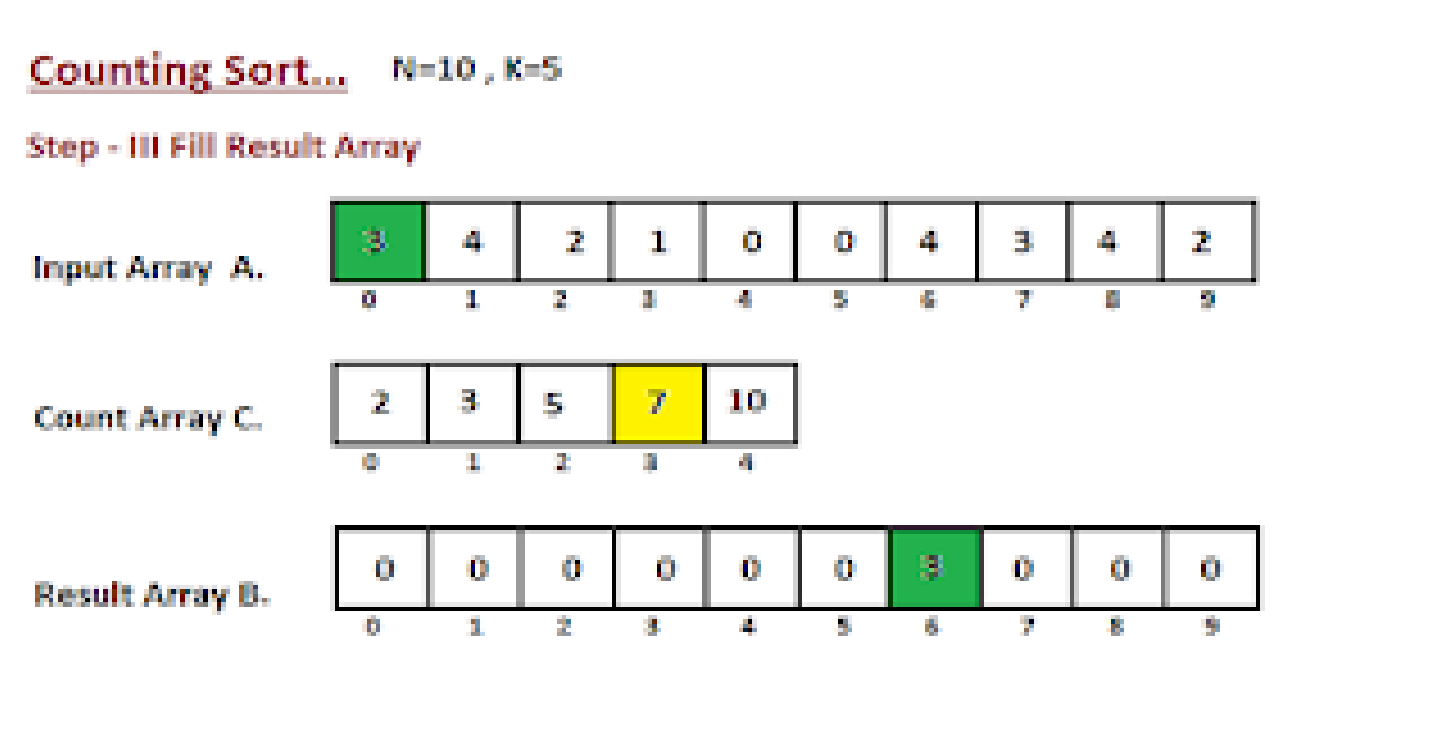
2	1	2	2	3
0	1	2	3	4

After Step -II Count Array Changes To As Shown Below.

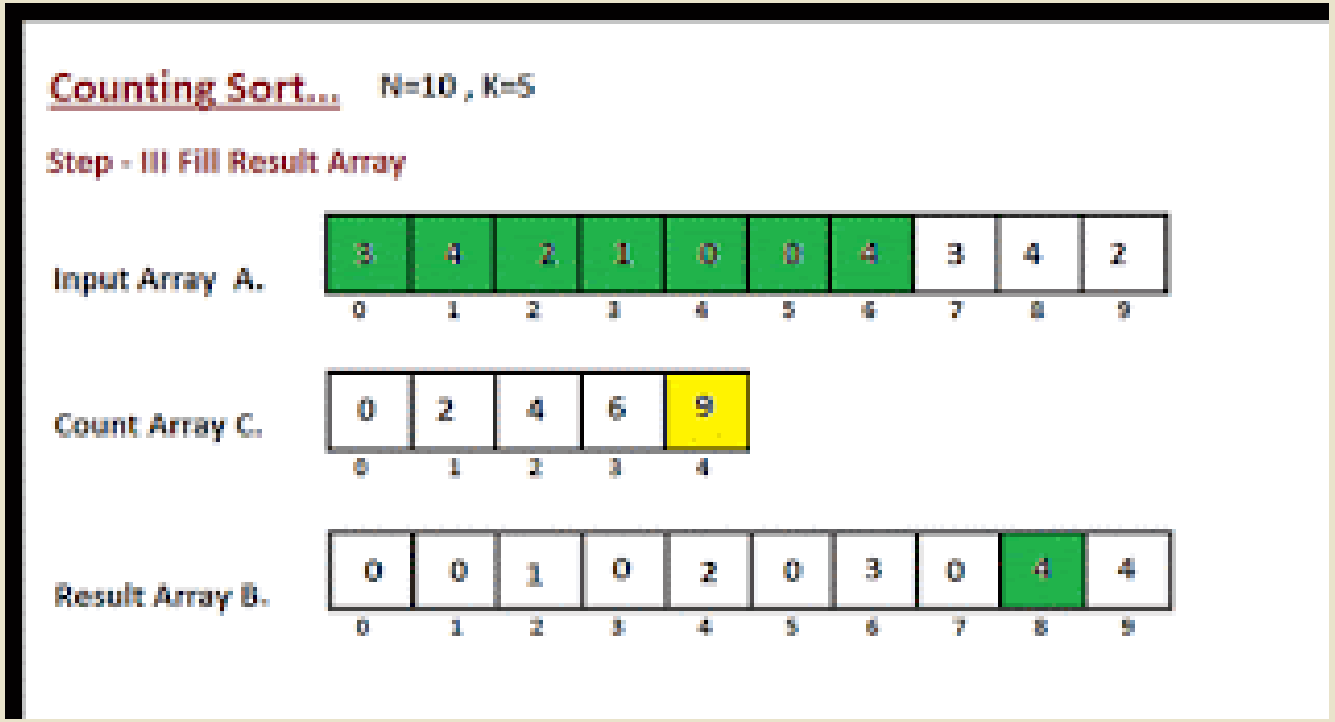
Count Array C.

2	3	5	7	10
0	1	2	3	4

مرتب سازی شمارشی



مرتب سازی شمارشی



مرتب سازی شمارشی

```
COUNTING-SORT(L: ARRAY; n, k: integer)
{
    // L is an array of size n
    // All the elements are in range [1, k]
    // C is an array with size of k
    for i = 1 to k do
        C[i] = 0;
    for i = 1 to n do
        C[L[i]]++;
    for i = 2 to k do
        C[i] = C[i] + C[i-1];
    // C[i] contains num of elements ≤ i
    for i = n downto 1 do {
        result[C[L[i]]] = L[i];
        C[L[i]]-- ;
    }
    for i = 1 to n do
        L[i] = result[i];
}
```

❊محدودیت : اعداد باید طبیعی باشند

مرتب سازی مبنایی

```
RADIX-SORT( L : ARRAY; n, k, b : integer )
{
    // L is of size n, with elements of size k
    // r is the radix of elements
    // Bucket is a hash table of size b
    var Bucket : HASH;
    for i = k downto 1 do
   {
       for j = 1 to n do
      {
            bucketnum = ith character of L[j];
            INSERT(Bucket[bucketnum], L[j]);
       }
       Collect all the elements from the buckets
       in order, and put them in list L;
   }
}
```

جمع بندی مرتب سازی ها

مرتب سازی	مقایسه ای / غیر مقایسه ای	درجا / برون جا	پایدار / ناپایدار	پیچیدگی زمانی متوسط	ویژگی خاص
مرتب سازی درجی Insertion sort	مقایسه ای	درجا	پایدار	$\theta(n^2)$	برای تعداد کم عناصر ، بسیار مناسب است
مرتب سازی حبابی Bubble sort	مقایسه ای	درجا	پایدار	$\theta(n^2)$	پیاده سازی های دیگری مانند مرتب سازی لرزشی دارد
مرتب سازی انتخابی Selection sort	مقایسه ای	درجا	پایدار	$\theta(n^2)$	ساده ترین و بدترین مرتب سازی مقایسه ای است.
مرتب سازی صدفی Shell sort	مقایسه ای	درجا	ناپایدار	$\theta(n^{\frac{3}{2}})$	پیاده سازی های بهتری از مرتبه ی $\theta\ (n^{6/5})$ و $\theta\ (n^{6/5})$ دارد
مرتب سازی هرمی Heap sort	مقایسه ای	درجا	ناپایدار	$\theta(n \log n)$	از ساختمان داده ی هرم استفاده می کند

جمع بندی مرتب سازی ها

مرتب سازی	مقایسه ای / غیر مقایسه ای	درجا / برون جا	پایدار / ناپایدار	پیچیدگی زمانی متوسط	ویژگی خاص
مرتب سازی ادغامی Merge sort	مقایسه ای	برون جا	پایدار	$\theta(n \log n)$	از جمله الگوریتم های تقسیم و غلبه است که به صورت بازگشتی پیاده سازی می شود. کار اصلی بر دوش تابع ادغام است.
مرتب سازی سریع Quick sort	مقایسه ای	درجا	ناپایدار	$\theta(n \log n)$	از جمله الگوریتم های تقسیم و غلبه است که به صورت بازگشتی پیاده سازی می شود. کار اصلی بر دوش تابع افراز است. سریع ترین الگوریتم مرتب سازی مقایسه ای است.

جمع بندی مرتب سازی ها

مرتب سازی	مقایسه ای / غیر مقایسه ای	درجا / برون جا	پایدار / ناپایدار	پیچیدگی زمانی متوسط	ویژگی خاص
مرتب سازی شمارشی Counting sort	غیر مقایسه ای	برون جا	پایدار	$\theta(n+k)$	استفاده از آن محدود به شرایطی است که عناصر از بازه ای متناهی و محدود باشند
مرتب سازی مبنایی Radix sort	غیر مقایسه ای	برون جا	پایدار	$\theta(kn+kr)$	در مرتب سازی اعداد و رشته ای از کاراکتر ها کاربرد دارد.

پایان