Сортирање

Поим

Сортирањето може да се каже дека претставува процес на преуредување на дадено множество објекти по некој однапред зададен критериум.
Кога пребаруваме по некој телефонски број во именик или збор во речник ние го користиме нашето искуство и знаење за тоа како се организирани податоците во именикот или речникот (алфабетски подредени т.е сортирани).

Ако податоците не се подредени во некој логички редослед пронаоѓањето некој податок претставува вистинска тешкотија или одзема многу време.
Целта на сортирањето е да се олесни и забрза пребарувањето на објектите.
За сортирањето може да се каже дека е една од најважните активности при обработката на податоците.

Сортирањето е проблем за кој постојат голем број разновидни алгоритми наречени алгоритми за сортирање кои меѓусебно се разликуваат по времето потребно за извршување како и потребните мемориски ресурси односно сложеност.

Формална дефиниција
Нека земеме една произволна низа $S=\{sk_1, sk_2, ..., sk_n\}$ составена од $n \geq 0$ елементи од некое универзално множество U.
Целта на сортирањето е да се преорганизираат елементите од низата S создавајќи некоја нова низа S' во која елементите од низата S ќе бидат подредени.
Но со што всушност се пртеставува подреденоста на елементите?.

Треба да дефинираме релација < која ќе претставува тотално подредување над елементите од множеството U на следниов начин:

Дефиниција:
Тотално подредување е релацијата '< 'дефинирана над елементите од некое универзално множество со следните атрибути:

1.За сите парови елементи (i,j) од UxU точно едно од следните тврдења е вистинито: ij.
2. За сите тројки (i,j,k) од UxUxU i i
(релацијата е транзитивна).

За да ги сортираме елементите од множеството S ние треба да најдеме пермутација

$P=\{p_1, p_2, ..., p_n\}$ така што за елементите од множеството S ке важи:
$sp_1 <= sp_2 <= ... <= sp_n$

Во суштина ние не сме заинтересирани за пермутацијата туку нашата цел е сортираната низа $S'=\{s_1', s_2', ..., s_n'\}$ за чии елементи важи
$s_i'=sp_i$ за $1<=i<=n$.

Алготитми за сортирање

Ке обработиме неколку алгоритми за сортирање организирани во следниве категории според главниот медот на кој се засновани:

- Сортирање со вметнување –(eng. Insertion sort)
- Сортирање со замена – (eng. Exchange sort)
- Сортирање со избор – (eng. Selection sort)
- Сортирање со мешање – (eng. Merge sort)
- Сортирање со распределба – (eng. Distribution sort)

Алгоритми за сортирање со вметнување(Insertion sort)

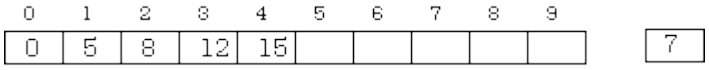
Основна идеја

Нека е дадена некоја низа од броеви која треба да се сортира.
Во секој чекор од алгоритмот постои дел од низата во кој елементите се сортирани (на почетокот се зема дека првиот елемент претставува сортирана низа), и исто така треба да се најде "точна" позиција на која треба да се вметне нов елемент во низата.

Ако низата се сортира во растечки редослед потребно е да се најде позиција во низата со вредност поголема од вредноста на елементот кој го вметнуваме.
Ако таква позиција нема тогаш позицијата на новиот елемент е после последниот елемент во сортираната низа.
Новиот елемент се вметнува на пронајдената позиција но притоа сите елементи од таа позиција до крајот на низата се поместуваат за едно место во десно за да се ослободи простор за елементот кој се вметнува.

Пример
Да претпоставиме дека во некој чекор од алгоритмот ја имаме следната ситуација:

A [0] = 0, A [1] =5, A [2] = 8, A [3] = 12, A [4] = 15



и 7 е вредноста на новиот елемент кој треба да го вметнеме.

Според сугестиите наведени претходно пребаруваме низ низата A[1],A[2],…A[i] додека не го пронајдеме првиот елемент со вредност поголема од 7 (во овој случај тоа е 8),индексот на тој елемент е позицијата на која треба да биде вметнат новиот елемент додека елементите A[4],A[3],A[2] мора да бидат поместени за една позиција во десно за да се ослободи место за новиот елемент.

После вметнувањето се добива следната ситуација:

0	1	2	3	4	5	6	7	8	9
0	5	7	8	12	15				

Алгоритам

Алгоритмот за реализација на овој метод се состои од следните чекори:

1.Претпостави дека првиот елемент од низата претставува сортирана низа.
2.Започни со следниот елемент од низата (несортиран дел).
3.Вметни го новиот елемент во сортираниот дел од низата на точната позиција .
4.Сортираниот дел од низата зголеми го за еден.Повторувај ја постапката се додека не се сортира целата низа.

Имплементација

Следниов програмски код напишан во програмскиот јазик Pascal покажува една имплементација на алгоритмот за сортирање со вметнување.

```
procedure insertionsort(var a:niza;n:integer);
                                    {зема за аргументи низа и должина на низата}
var i,j,temp:integer;
begin
    For i:=2 to n do
    begin
     j := i;
       temp := a[j];
       While (j>1) and (a[j-1]>temp) do
            begin
             a[j] := a[j-1];
             Dec(j);
            end;
    a[j] := temp;
   end;
end;
```

Анализа за сложеност

Најдобар случај : ако низата е веќе сортитана тогаш внатрешниот while циклус нема да се изврши,а надворешниот for циклус ке се изврши n-1 пати од каде се добива сложеност O(n).
Најлош случај:ако низата е сортирана во обратен редослед тогаш имаме максимален број на извршување на внатрешниот while циклус од каде се добива сложеност O().

Алгоритмот кој го анализиравме користи линеарно пребарување за позицијата на која вршиме вметнување, но бидејќи може да се забележи дека целната подниза во која вршиме вметнување е веќе сортирана , може да се искористи побрз алгоритам за пронаоѓање на местото на кое треба да се вметни следниот елемент.

Идеален алгоритам за пребарување е алгоритмот на бинарно барање од каде и алгоримот за сортирање го добива името алгоритам за сортирање со бинарно вметнување.

Значи во оваа модифицирана верзија на алгоритмот за сортирање со вметнување ние всушност користиме бинарно пребарување (поефикасен алгоритам ,со логаритамската комлексност) за позицијата на која треба да вршиме вметнување,
додека останатиот дел од алгоритмот е потполно идентичен.

Останува непроменет бројот на поместувања на податоците во десно за ослободување на место за вметнување.

Ефикасноста на овој алгоритам во најдобар случај е O(nlogn),
додека во најлош случај O().

Следниот програмски код напишан во програмскиот јазик Pascal е една имплементација на алгоритмот за сортирање со бинарно вметнување.

```
procedure binaryinsertionsort(var a:niza;n:integer);
var
    temp,i,j,l,r,m: integer;
begin
 for i:=2 to n do
  begin
  temp := a[i];

     l := 1;
     r := i-1;
      while l<=r do
        begin
          m := (l+r) div 2;          {koristime binarno baranje za pozicijata na koja}
           if  temp < a[m] then   {treba da go vmetneme noviot element}
                 r := m-1
                   else
                 l := m+1
        end;{while}

     for j:= i-1 downto l do
     a[j+1]:= a[j];
  a[l]:= temp;
 end;{for}
end;{procedure}
```

Сложеноста на овој алгоритам се разгледува во однос на пребарување за позицијата на новиот елемент и во најдобар случај е O(nlogn).

Алгоритми за сортирање со замена (eng Exchange Sort)

Втората класа на алгоритми за сортирање кои ке ги разгледаме се алгоритмите за сортирање со замена на пар елементи се додека низата не е сортирана.

Ќе ги разгледаме методите:

- Метода на меурче (eng Bubble Sort)
- Брзо сортирање (eng Quick Sort)

Сортирање со метода на меурче (eng Bubble Sort)

Основна идеја

За да се сортира низата $a_1, ..., a_n$ со оваа метода потребно е да се направат n-1 премини во низата.Во секој премин се споредуваат парови соседни елементи и ако има потреба т.е. ако елементите кои се споредуваат не се во бараниот редослед , тие си ги заменуваат местата.

После првиот премин најмалиот елемент од низата ќе дојде на првата позиција од низата (избива на површината како меурче во чаша вода). После k премини над низата последните k елементи од низата се на "точната" позиција и не треба повеќе да се разгледуваат. За да се осигураме дека низата е сортирана комплетно потребни се n-1 премини иако таа може да биде сортирана и порано.

Пример
Да се сортира низата 234,77,0,113,404,60 со помош на методата на меурче:

```
Премин број 1:
   77  234    0  113  404   60
   77    0  234  113  404   60
   77    0  113  234  404   60
   77    0  113  234  404   60
   77    0  113  234   60  404

 Број на замени во овој премин = 4

 Премин број 2:
   0   77  113  234   60  404
   0   77  113  234   60  404
   0   77  113  234   60  404
   0   77  113   60  234  404
  Број на замени во овој премин=2

Премин број 3:

    0  77  113  60  234  404
    0  77  113  60  234  404
    0  77  60  113  234  40
Број на замени во овој премин=1

Премин број 4:
  0  77  60  113  234  404
  0  60  77  113  234  404
Број на замени во овој премин=1
```

После овој премин низата е веќе сортирана иако алгоритмот за да го провери тоа мора да направи уште еден премин над елементите.

Алгоритам

1.Повторувај ги чекорите од 2 до 4 се додека се можни замени
2. Од еден до должината на низата минус еден извршувај чекор 3
3.Ако соседните елементи (првиот со вториот,вториот со третиот итн.)не се во бараниот редослед замени им ги местата.
4.Должината на низата над која се извршува алгоритмот намали ја за еден.
5.Ако се можни замени оди на чекор 2.
Овој алгоритам гарантира дека после секој премин сигурно еден елемент од низата доаѓа на точната позиција.

Имплементација

Овој алгоритам е многу едоставен за имплементација и разбирање.
Следниов програмски код прикажува една негова имплементација во програмскиот јазик Pascal.

```
Procedure bubblesort(var a:niza;n:integer);
var temp,i,j: integer;
begin
    for i:= n downto 2 do
       begin
           for j:= 2 to i do
              begin
                  if (a[j-1] > a[j]) then
                        begin
                          temp:= a[j-1];
                          a[j-1]:= a[j];
                          a[j]:= temp;
                        end;{if}
              end;{for j}
       end;{for i}

End;
```

Анализа

Овој алгоритам има квадратна сложеност $O(n^2)$ бидејќи и во најдобар случај кога низата е сортирана потребни се истиот број на споредби за да со сигурност се потврди дека низата е сортирана.Можни се некои модификации над првичниот алгоритам со кои се добива на забрзување со тоа што ако низата се сортира пред сите можни премини алгоритмот ќе ја открие таквата состојба и ќе го прекине понатамошното извршување.

Метод на брзо сортирање (eng QuickSort)

Овој алгоритам е од типот раздели па владеј (eng divide-and-conquer).

Еден алгоритам е од типот раздели па владеј ако проблемот го решава така што најпрвин го разделува на помали подпроблеми,а потоа решавајќи ги подпроблемите и комбинирајќи ги нивните решенија доаѓа до бараното решение т.е решението на основниот проблем.

Основна идеја

Нека a_1, ..., a_n е дадена низа која треба да се сортира.
Методот на брзо сортирање проблемот на сортирањето го решава на следниот начин:

-еден елемент од низата се прогласува за главна точка-пивот(eng pivot),се пронаоѓа и поставува на "точната" позиција на која треба да стои.
-елементите од низата се делат на две поднизи во првиот дел одат елементите кои се помали од главната точка ,а во вториот дел оние кои се поголеми.
-алгоритмот рекурзивно се применува за двете поднизи се додека не се дојде до интервал со должина еден.

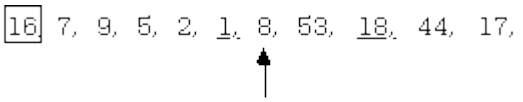
Пример:
Да се сортираат елементите 16,7,9,44,2,18,8,53,1,5,17 со помош на методата за брзо сортирање.Алгоритмот се изведува со помош на користење на два индекси иницијално поставени едниот на почеток а другиот на крајот од низата.

Го избираме првиот елемент #1(16) како главна точка и продолжуваме со пребарување,откриваме дека четвртиот елемент #4(44) е првиот поголем елемент од 16 и застануваме тука ,сега започнуваме со елементот #11(17) на крајот од низата и работиме наназад,елементот #10(5) е првиот елемент помал од 16 и тука вршиме смена со елементот на четвртата позиција #4(44).
Ја добиваме следната ситуација:

16 7, 9, 5, 2, 18, 8, 53, 1, 44, 17,

Подвлечените елементи се тие два елементи кои си ги заменија местата се разбира на индексите каде што застанавме со броењето од почетокот и крајот на низата.

Продолжуваме,следните два елементи кои треба да си ги заменат местата затоа што се наоѓаат на погрешни страни од низата се елементите #6(18) и #9(1)со што добиваме:

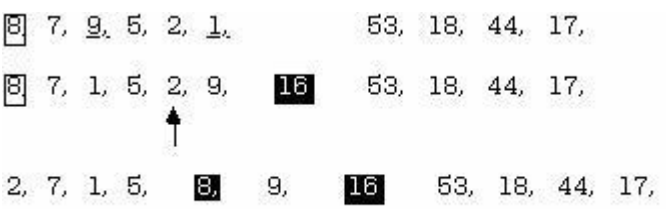


Продолжуваме следниот поголем елемент од пивотот е елементот #8(53) но тука веќе индексите кои ни покажуваат до каде сме стигнале се изедначуваат со што не се можни повеќе замени па пивотот се поставува на точната позиција (индекс минус еден) вршејќи замена со елементот кој е на таа позиција со што се добива:

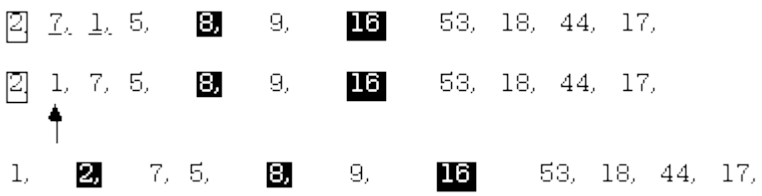
8, 7, 9, 5, 2, 1, 16, 53, 18, 44, 17,

Сега низата е поделена на две поднизи,а елементот 16 е на "точната" позиција и не треба повеќе да се разгледува (оттука името главна точка).Размислувајќи рекурзивно истата постака може да се примени за двете поднизи со што би се сортирала целата низа.

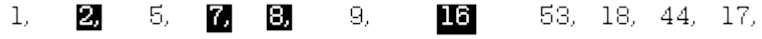
Повторувајќи ја постапката за левата подниза добиваме:



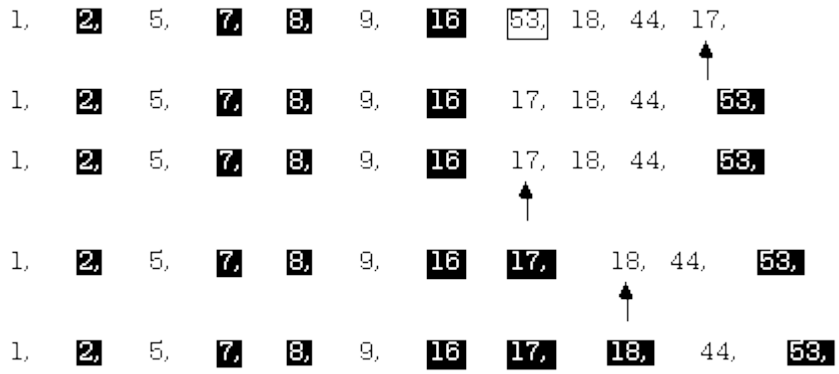
Повторувајќи ја постапката за левата подниза околу 8 , се добива:



Левата подниза со главна точка 2 е сортирана(еден елемент).Кога и десната подниза со главна точка 2 (два елементи) ке се исортира добиваме:



Со повторување на истата процедура ќе се добие следната слика:



Алгоритам

Алгоритмот за реализација на овој метод се состои од следните чекори:

1.Постави го првиот елемент од низата за главна точка-пивот.
2.Најди ја точната позиција на тој елемент и постави го таму.
3.Распореди ги елементите од низата т.ш. елементите кои се пред
главната точка-пивот се помали,а тие после неа поголеми од неа.
4.Повторувај ги чекорите рекурзивно за левата подниза се додека има повеќе од еден елемент.
5.Повторувај ги чекорите рекурзивно за десната подниза се додека има повеќе од еден елемент.

Алгоритмот гарантира дека во секој чекор еден елемент ке дојде на вистинското место.

Имплементација:

Следниот програмски код напишан во програмскиот јазик Pascal е една имплементација на алгоритмот за брзо сортирање.

```
procedure quicksort(var a:niza;lo,hi:integer);
procedure Sort(l, r: integer);
 var
     i, j, x, y: integer;
 begin
    i := l;
    j := r;
    x := a[l];
       repeat
               while a[i] < x do i := i + 1;
               while x < a[j] do j := j - 1;
                 if i <= j then begin
                       y := a[i];
                       a[i] := a[j];
                       a[j] := y;
                       i := i + 1;
                       j := j - 1;
                 end;
       until i > j;
if l < j then Sort(l, j);
if i < r then Sort(i, r);
end;

begin
Sort(lo,hi);
end;
```

Анализа

Најдобар случај за овој алгоритам има кога во секој чекор од рекурзијата од поделбата на низата настануваат точно два дела со еднаква должина,тогаш длабочината на рекурзијата е log(n).Од тука следува дека комлексноста е O(nlog(n)).

Најлош случај има кога во секој чекор на рекурзијата настанува небалансирана поделба,имено кога едната подниза се состои само од еден елемент а другата од останатите,тогаш длабочината на рекурзијата е n-1 па за сложеноста на алгоритмот се добива $O(n^2)$. Можни се модификации за забрзување на овој алгоритам и тоа во делот при изборот на елемент пивот.

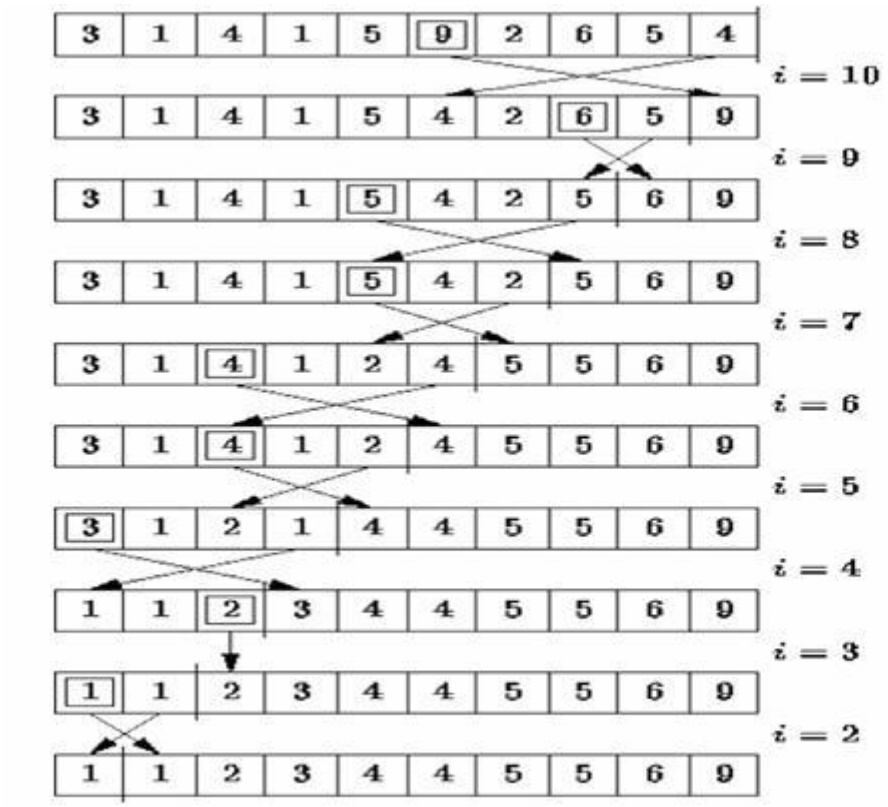
Алгоритми за сортирање со избор (eng Selection Sort)

Идејата за решавање на проблемот на сортирање кај овие алгоритми е следната:
во секој чекор елементот кој се додава во сортираната низа се избира од останатите елементи.
Кај овие видови алгоритми вметнувањето нови елементи се врши од еден крај на низата што ги прави различни од алгоритмите за сортирање со вметнување каде вметнувањето се врши на произволна позиција.

Основна идеја

За реализација на овој алгоритам потребни се n-1 премини во низата. Во секој чекор на алгоритмот вршиме линеарно барање низ несортираниот дел од низата барајќи ја позицијата на најголемиот елемент.Тој елемент го поставуваме на крајната позиција вршејќи замена со елементот кој се наоѓа на таа позиција,после секој чекор должината на низата над која се извршува алгоритмот ја намалуваме за еден.

Пример
Да се сортира низата 3,1,4,1,5,9,2,6,5,4 со помош на методата за сортирање со избор.При секој премин го избираме најголемиот елемент и го поставуваме на точната позиција вршејќи замена со елементот кој е на таа позиција.



Алгоритам

Алгоритмот може да се опише со следните чекори:

1. Со линеарно барање најди го максимум на елементите од низата.
2.Замени ги крајниот елемент од низата и максумумот на низата
3.Повтори ја постапката за преостанатите n-1 елементи,потоа за n-2 итн се додека не остане само еден т.е. најмалиот елемент.

Имплементација

Следниот програмски код напишан во програмскиот јазик Pascal е една имплементација на алгоритмот сортирање со избор.

```
Procedure selectionsort(var a:niza;n:integer);
Var
        i,j,temp,m: Integer;
Begin
    for i:=n  downto 2 do
      begin
        m:=1;
            for j:=1 to i do
              begin
            if (a[j] > a [m]) then
                    m:=j;
            end;

                    temp:=a[i];
                    a[i]:=a[m];
                    a[m]:=temp;

        end;
end;
```

Анализа

Овој алгоритам има константна сложеност и тоа $O(n^2)$ бидејќи секогаш се извршуваат истиот број на споредби и замени.

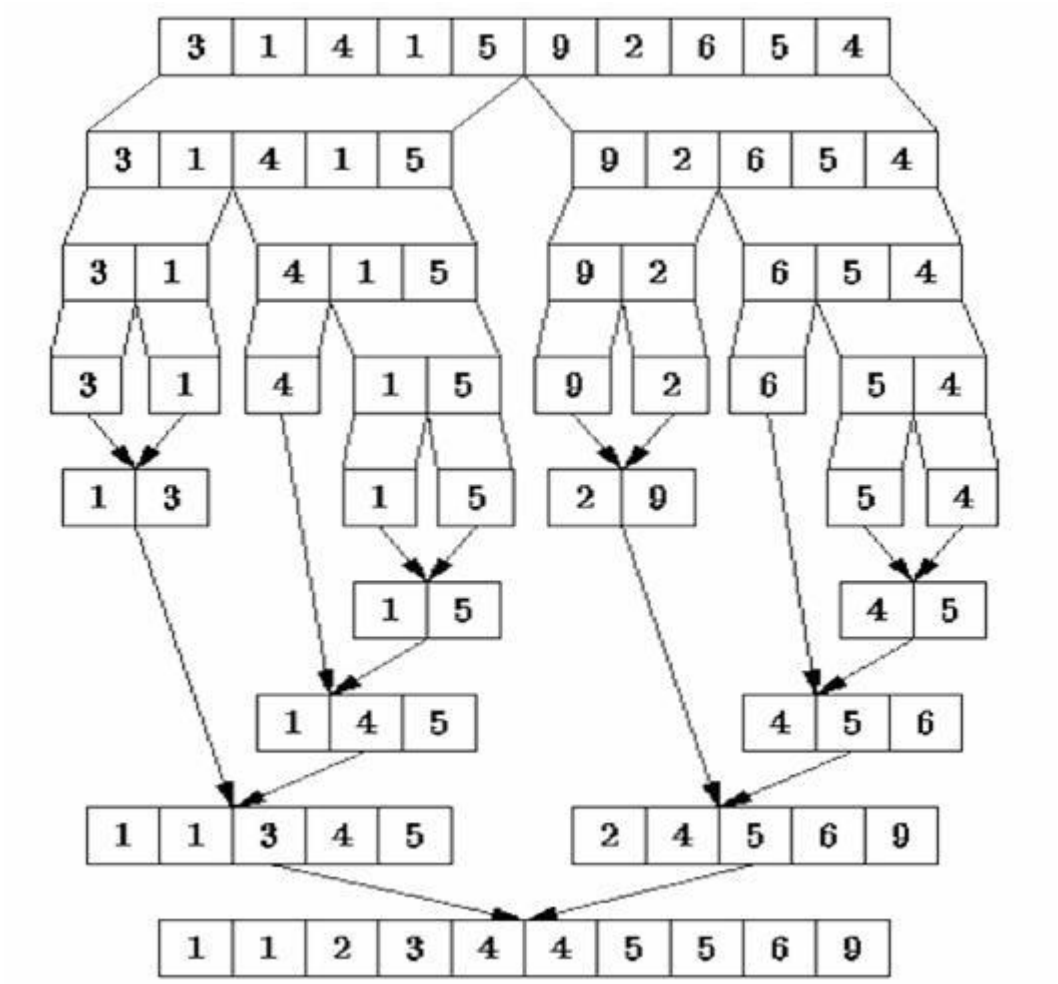
Друг алгоритам од оваа класа на алгоритми за сортирање со избор е алгоритмот за сортирање со Heap.Овој алгоритам има оптимална сложеностO(nlog(n)).

Алгоритми за сортирање со мешање (eng Merge Sort)

Основна идеја

Идејата за сортирање кај овој алгоритам е слична на идејата на алгоритмот за брзо сортирање.И овој алгоритам е од типот раздели па владеј,најпрво низата која треба да се сортира ја разделува на половина т.е. на две поднизи и потоа секоја подниза се сортира независно, на крај двете сортирани поднизи се мешаат со што се добива оригиналното решение.

Пример
Со помош на метод на претопување да се сортира низата 3,1,4,1,5,9,2,6,5,4.
Во секој чекор низата се разделува на половина и рекурзивно се повикува процедурата за сортирање за левата и десната подниза,и потоа процедурата за мешање.



Алгоритам

Алгоритмот може да се опише со следните чекори:

1.Раздели ја низата на две поднизи со должина n/2 ,n/2
2.Рекурзивно сортирај ја секоја од двете поднизи.
3.Измешај ги сортираните поднизи за да се добие крајниот резултат.

Базен случај за алгоритмот е подниза со точно еден елемент и таа се зема за сортирана.

Имплементација

Постојат повеќе начини за имплементација на процедурата за мешање (merge) но таа најчесто се имплементира на следниот начин:

Двете поднизи кои се мешаат најпрво се копираат во една помошна низа,потоа двете поднизи сега составен дел од помошната низа со помош на индекси кои покажуваат кои се нивни делови се преминуваат и помалиот елемент од поднизите се копира назад во почетната низа.Може да настане ситуација така што едниот индекс да дојде до крајот а другиот не, што означува дека во едната пониза има уште елементи и тогаш природно е преостанатите елементи да се ископираат на крајот од почетната низа.

Следниот програмски код напишан во програмскиот јазик Pascal е една имплементација на алгоритмот за сортирање со мешање.

```
Procedure mergesort(Var a: niza; lo, hi: Integer);
Label return;
Var end_lo,k,mid,start_hi: Integer;
   T: integer;
Begin
if lo>=hi then goto return;
 mid := (lo + hi) Div 2;
 {Podeli ja listata na dve podnizi i sortiraj gi rekurzivno}
 mergesort(a, lo, mid);
 mergesort(a, mid + 1, hi);
 {Izmesaj gi dvete podnizi vo edna sortirana}
 start_hi := mid + 1;
 end_lo := mid;
 while (lo <= end_lo) and (start_hi <= hi) do
 begin
  if a[lo] < a[start_hi] then
   Inc(lo)
  else
  begin
   T := a[start_hi];
   for k := start_hi - 1 Downto lo do a[k+1] := a[k];
   a[lo] := T;
   Inc(lo);
   Inc(end_lo);
   Inc(start_hi)
  end
 end;
return: End;
```

Анализа

Процедурата за мешање побарува најмногу 2n чекори (n чекори за копирање на низата во помошната низа и n чекори за враќање на елементите назад),од каде за комплексноста на овој алгоритам добиваме :

$T(n) \leq 2n + 2\ T(n/2)$ и

$T(1) = 0$

од каде со решавање на оваа рекурентна равенка се добива

$T(n) \leq 2n \log(n)\ \varepsilon\ O(n \log(n))$.

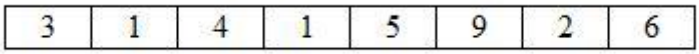
Алгоритми за сортирање со распределба (eng Distribution Sort)

Главна карактеристика на овие алгоритми е тоа што тие не користат споредби за да извршат сортирање.

Bucket Sort

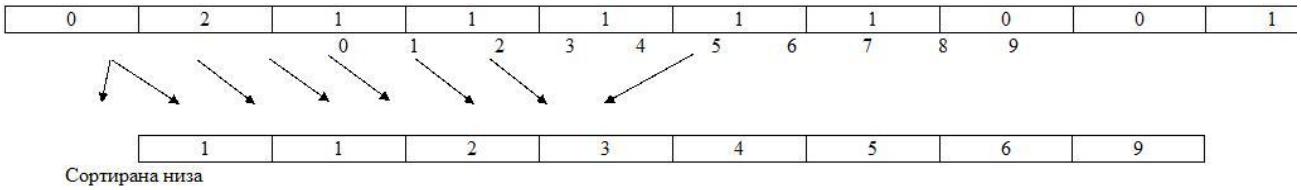
Овој алгоритам е можеби наједноставниот алгоритам за сортирање со распределба. Суштинското барање на овој алгоритам е тоа што големината на множеството од каде што се елементите кои ги сортираме е мало и константно.

На пример да претпоставиме дека треба да ги сортираме елементите од множеството {0,1,... n-1} т.е. подмножество од множеството цели броеви.
Овој алгоритам користи n бројачи, i-от бројач чува број на појавувања на i-от елемент од множеството.
Пример:
Да се сортира низата 3,1,4,1,5,9,2,6 со методот BucketSort.



Почетна низа

Низа од бројачи со должина колку максималниот елемент плус еден



Сортирана низа

Имплементација

Следниот програмски код напишан во програмскиот јазик Pascal е една имплементација на овој алгоритмот за сортирање.

```
procedure bucketsort(var a:niza;n:integer;m:integer);
            {argumenti: a-niza,n-broj na elementi,m-maksimalen element na nizata}
 var bukets:niza;     { treba array[1..maximalen_od_nizata] of integer;}
      pom,i,j,k:integer;
begin
for j:=0 to m do {inicijalizacija}
begin  bukets[j]:=0; end;
for i:=1 to n do
    begin
      pom:=a[i];
      bukets[pom]:=bukets[pom]+1;
    end;



  i:=0; {generiranje na nizata}
  for j:=0 to m do
   begin
      for k:=1 to bukets[j]  do
       begin
          i:=i+1;
          a[i]:=j;
       end;
   end;
end;
```

Анализа

Времето потребно за извршување на овој алгоритам е од ред O(m+n).