

int binarySearch(int a[], int item, int low, int high)

{

if (high <= low)

return (item > a[low]) ? (low + 1) : low;

int mid = (low + high) / 2;

if (item == a[mid])

return mid + 1;

if (item > a[mid])

return binarySearch(a, item, mid + 1, high);

return binarySearch(a, item, low, mid - 1);

}

void insertionSort(int a[], int n)

{

int loc, j, k, selected;

for (int i = 1; i < n; ++i)

{

j = i - 1;

selected = a[i];

// find location where selected

//should be insertеd

loc = binarySearch(a, selected, 0, j);

// Shift right all elements after location

while (j >= loc)

{

a[j + 1] = a[j];

j--;

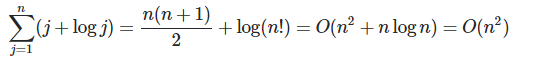
}

a[j + 1] = selected;

}

}

**Прилага се същият алгоритъм за сортиране чрез вмъкване, но вместо със сравнения на текущия елемент със всеки от предхождащите (вече подредени) се използва двоично търсене (в подредената част), за да се определи на коя позиция да се вмъкне. Сложността остава n^2, въпреки намаляването на броя на сравненията (от n на logn), тъй като преместванията на елементите остава n в най-лошия случай.**

**За j-ти element ще се извършат около log j сравнения и в най-лошия случай около j отмествания вдясно. Следователно извършваме по-малко сравнения, но отново работа с линейна сложност на всяка итерация. Това е и причината сложността да не се променя: **