Jump Searching

Jump-Sıçrayış axtarış methodu

---Jump searching nədir?



Bu method massivdəki lazım olan elementi tapmaq üçün işlədilir , başqa axtariş methodlarından fərqi sabit addimlarla , əsasəndə bir neçəlik addim tulanaraq özü üçün bir müqayisə araliği yaradir . Axtarilan element bu araliqdaa olmasa , bu araliğı birbaşa atlayır.

Addımın böyüklüyü massivdəki elementlərin sayının kök altısıdır .

---nümünə

Bizə verilmiş [] = {0, 1, 1, 2, 3, 5, 8, 13, 21, 55, 77, 89, 101, 201, 256, 780} bu massivdə 77 elementin axtarırıq

1 . Əvvəlcə siçrayiş vahididni təyin edək

m = √n = 4 n-elementlərin sayı , massivin uzunluğu

2.Sıradaki ilk element başlangiç sayıb 4cü element daxil olmaqla ilk araliği seçirik





Başlangıç elementinin şərti A[0] != item && A[0]<item

3.Element yuxardaki araliqda yerləşmədi deyə bu araliği keçirik , bəyaqkı aralığin son elementi ikinci araliğimizin başlangiçı sayırıq ,nəticə olaraq elementimiz bu araliqdan böyük olduğu üçün keşirik A[6] != item && A[0] <item şəriı bir daha pozulur .



4.Yenə alinmadi,Yuxardaki prossesləri bir daha təkrarlıyırıq



5.Ən sonda yerləşdiyi araliğı tapirıq , əvvəlcə başlangic olan 55 == 77 deyə yoxluyuruq əgər şərt ödənmirsə novbəti indexdəki elementə baxiriq ,və nəhayət tapırıq



----Time Complexity

Best case (Ən yaxşi halda) O(1)

Average (Orta) O(√ n)

Worst case (Ən pis halda) O(√ n)

---Üstünlüyü

1.Axtarış matorları açar sözü milyardlarca data arasindan bu methoda görə tapir siyahılıyır

2.Video oyunlarinda , oyuncunun hərəkətini optimizə etmək üçün işlədilir

3.Böyük datalarla işləmək üçün idealdir , həm sürət həmdə yaddaş performansi üstündür

--kodu

public class JumpSearchAlgorithm

{

public static void Main(string[] args)

{

int[] arr = new int[] { 10, 20, 30, 40, 50, 60, 70, 80, 90,100 ,110 ,120 ,130 ,140, 150, 160, 170,};

int ele = 130;

int foundIndex = jumpSearch(arr, ele);

System.Console.Out.WriteLine(foundIndex > 0 ? "Found at index : " + foundIndex :

"Element Not Found");

}

public static int jumpSearch(int[] arr, int ele)

{

int prev = 0;

int n = arr.Length;

int step = (int)System.Math.Floor(System.Math.Sqrt(n));

//loop until current element is less than the given search element

while (arr[System.Math.Min(step, n) - 1] < ele)

{

prev = step;

step += (int)System.Math.Floor(System.Math.Sqrt(n));

if (prev >= n)

{

return -1;

}

}

//perform linear search prev index element to given element

while (arr[prev] < ele)

{

prev++;

if (prev == System.Math.Min(step, n))

{

return -1;

}

}

// Return index if element is found

if (arr[prev] == ele)

{

return prev;

}

return -1;

}

}