**Analiza complexitate timp – metoda BingoSort**

**Lab 12 – Aplicatie InchiriereFilme**

**- Mihaita Tudor 214/2 –**

Metoda aleasa pentru studiul complexitatii din punct de vedere al timpului de executie este algoritmul de BingoSort, utilizat in rapoartele aplicatiei, algoritm care nu prezinta o complexitate timp constanta:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Let’s consider the following array as an example:****arr[]   = [5, 4, 8, 5, 4, 8, 5, 4, 4, 4]***   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **5** | **4** | 8 | 5 | 4 | 8 | 5 | 4 | 4 | 4 |   ***Step 1:****Find the smallest value which is called a Bingo Element. Smallest element = 4*  ***Step 2:****Shift all the elements of the array to their correct position which is equal to the Smallest element by swapping the position of Elements.*   * ***First Pass:***  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **4** | **5** | 8 | 5 | 4 | 8 | 5 | 4 | 4 | 4 |  * ***Second Pass***  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 4 | **4** | 8 | 5 | **5** | 8 | 5 | 4 | 4 | 4 |  * ***Third Pass****:*  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 4 | 4 | **4** | 5 | 5 | 8 | 5 | **8** | 4 | 4 |  * ***Fourth Pass:***  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 4 | 4 | 4 | **4** | 5 | 8 |  | 5 | 8 | **5** | 4 |  * ***Fifth Pass:***  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 4 | 4 | 4 | 4 | **4** | 8 | 5 | 8 | 5 | **5** |   ***Step 3:****Similarly find the next Bingo Element (or smallest element) and shift the elements to their correct position that are equal to Bingo Element.****Next Bingo Element = 5***   * ***Sixth pass:***  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 4 | 4 | 4 | 4 | 4 | **5** | **8** | 8 | 5 | 5 |  * ***Seventh Pass:***  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 4 | 4 | 4 | 4 | 4 | 5 | **5** | 8 | **8** | 5 |  * ***Eighth Pass:***  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 4 | 4 | 4 | 4 | 4 | 5 | 5 | **5** | 8 | **8** |   *Finally, Our array has been arranged in the increasing order as you can see above* |

* **Algoritmul BingoSort functioneaza astfel:**
* In cazul unei sortari crescatoare, se determina valoarea celui mai mic si celui mai mare element din lista initiala, nesortata, elementul minim curent fiind cunoscut drept **elementul Bingo**.
* Se creeaza o copie a elementului maxim pentru a putea utiliza variabila **nextBingo** in identificarea urmatorului element minim, mai mare decat elementul minim curent, si o variabila **nextElementPosition** ce reprezinta pozitia pe care se vor muta elementele egale cu elementul Bingo curent.
* Se ruleaza o structura repetitiva de tip while care, cat timp elementul curent Bingo este mai mic decat valoarea maxima din lista, va muta pe pozitia corespunzatoare din lista sortata toate elementele egale cu elementul Bingo curent, actualizand totodata nextBingo pentru a se continua interschimbarea elementelor in urma unei parcurgeri complete a listei cu elementul Bingo curent.
* In cazul unei sortari descrescatoare, algoritmul functioneaza identic, singura diferenta fiind data de comparator, care va inversa conditia de comparare in functie de parametrul reverse dat, atribuind astfel elementului Bingo curent valoarea maxima din lista de fiecare data si mutand dupa acesta toate aparitiile elementului corespunzator, continuandu-se astfel pana elementul Bingo curent ajunge la valoarea elementului minim din lista.
* **Analizarea cazurilor de complexitate timp:**

|  |  |
| --- | --- |
| **Algortim** | **Complexitate** |
| **# metoda de determinare a elementului minim din lista si a elementului maxim**  **def minMax(lista, key, reverse):  min = max = lista[0]  for i in range(1, len(lista)):  if cmp(lista[i], min, key, reverse):  min = lista[i]  if cmp(max, lista[i], key, reverse):  max = lista[i]  return min, max**  **# algoritmul de sortare BingoSort, ce va primi drept parametri lista nesortata key-ul(atribut) care va reprezenta criteriul de sortare, si reverse, determinand comparatorul pentru o sortare crescatoare sau descrescatoare**  **def BingoSort(lista, key, reverse):  bingo, nextBingo = minMax(lista, key, reverse)  largestElement = nextBingo  nextElementPos = 0while cmp(bingo, nextBingo, key, reverse):  startPos = nextElementPos  for i in range(startPos, len(lista)):  if lista[i] == bingo:  lista[i], lista[startPos] = lista[startPos], lista[i]  nextElementPos += 1  elif cmp(lista[i], nextBingo, key, reverse):  nextBingo = lista[i]  bingo = nextBingo  nextBingo = largestElement   return lista** | Pentru definirea cazurilor de complexitate, vom utiliza urmatoarele variabile: n = numarul de elemente din lista; m = numarul de valori distincte din lista; k = numarul(mediu) de aparitii a unei valori distincte in lista data.   * **Best-Case Complexity:**   Lista contine doar elemente egale. In acest caz, se va executa o singura data metoda minMax, va determina valorile variabilelor Bingo si nextBingo, iar cum acestea vor fi egale, nu se va mai intra in structura repetitiva while.  T(n) =  SAU:  Lista contine un numar de m valori distincte, insa acestea sunt distribuite uniform, numarul de pasi efectuat de a doua structura repetitiva din interiorul while-ului micsorandu-se astfel succesiv, cu fiecare parcurgere cu valoare curena a elementului Bingo.  T(n) =   * **Worst-Case Complexity:**   Lista contine doar valori distincte, deci n=m, iar k=1, fiecare valoare fiind unica.  T(n) =   * **Average-Case Complexity:**   In lista exista m elemente distincte ce se pot gasi in grupe de 1,2,…,n in cele n elemente ale listei. Astfel, in functie de numarul de aparitii al fiecarui element distinct din lista, se va modifica numarul de pasi efectuati de structura for pentru mutarea elementelor egale cu elementul Bingo curent.  T(n) =   * **Overall Complexity:** |