**Insertion sort**

تعد هذه الخوارزمية من أبسط خوارزميات الفرز فهي تعمل بشكل مشابه للطريقة التي نقوم فيها بترتيب أوراق اللعب في يدنا.

تفترض أن البطاقة الأولى مرتبة بالفعل وباقي البطاقات غير مرتبة إذا كانت البطاقة أكبر من البطاقة التي تسبقها يتم نقل البطاقة موضع واحد إلى اليمين

أما إذا كانت أصغر يتم مقارنتها مع باقي العناصر التي تسبقها وبنفس الطريقة يتم أخذ بطاقات أخرى غير مرتبة وتقوم بمقارنتها مع جميع البطاقات التي تقع على يمينها وذلك لوضعها في مكانها المناسب

**ألية العمل :**

**لنفترض أننا نريد ترتيب هذه المصفوفة ترتيبا تصاعديا**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **9** | **5** | **1** | **4** | **3** |

العنصر الأول في المصفوفة يعتبر مرتب بالفعل نأخذ العنصر الثاني ونبدأ بمقارنته مع العنصر الأول إذا كان العنصر الأول أكبر نقوم بوضع العنصر الحالي على يمين العنصر الأول

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **9** | **5** | **1** | **4** | **3** |

الان أول عنصرين مرتبين نأخذ العنصر الثالث ونقوم بمقارنته مع جميع العناصر التي تقع على يساره ووضعه على يسار العناصر الأصغر منه إذا لم يكن هناك عناصر أصغر منه يبقى في مكانه

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5** | **9** | **1** | **4** | **3** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **5** | **9** | **4** | **3** |

وبشكل مشابه ضع كل عنصر غير مرتب في موضعه الصحيح

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **4** | **5** | **9** | **3** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **3** | **4** | **5** | **9** |

**مرتبة**

**التعقيد الزمني للخوارزمية :**

في الحالة الأسوء يكون التعقيد الزمني o(n2)

أما في الحالة الأفضل التي تكون فيها عناصر المصفوفة مرتبة بالفعل يكون التعقيد الزمني o(n)

البرنامج:

// C++ program for insertion sort

#include <bits/stdc++.h>

using namespace std;

/\* Function to sort an array using insertion sort\*/

void insertionSort(int arr[], int n)

{

    int i, key, j;

    for (i = 1; i < n; i++)

    {

        key = arr[i];

        j = i - 1;

        /\* Move elements of arr[0..i-1], that are

        greater than key, to one position ahead

        of their current position \*/

        while (j >= 0 && arr[j] > key)

        {

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = key;

    }

}

// A utility function to print an array of size n

void printArray(int arr[], int n)

{

    int i;

    for (i = 0; i < n; i++)

        cout << arr[i] << " ";

    cout << endl;

}

/\* Driver code \*/

int main()

{

    int arr[] = { 12, 11, 13, 5, 6 };

    int n = sizeof(arr) / sizeof(arr[0]);

    insertionSort(arr, n);

    printArray(arr, n);

    return 0;

}

// This is code is contributed by rathbhupendra