Selection Sort

Định Nghĩa

Selection sort là dạng sắp xếp cơ bản của các thuật toán để sắp xếp lại các giá trị chưa được sắp xếp bằng cách liên tục tìm số nhỏ nhất và sắp xếp lại bảng chứa data đó theo giá trị tăng dần

Selection sort dùng trong để kiểm tra các giá trị trong bảng data để cho mọi giá trị đều được sắp xếp theo trình tự (Từ nhỏ nhất đến lớn nhất). Khác với các thể loại sắp xếp khác thì Selection sorting dùng rất ít bộ nhớ để thực hiện quá trình



Khi máy tính thực hiên Selection sort trong 1 bảng chứa data chưa được sắp xếp. Nó sẽ đi qua từng element trên bảng theo trình tự và tìm ra số nhỏ nhất trong bảng chứa data đó bằng cách so sánh các giá trị của các số trong bảng chứa data với nhau. Sau khi tìm được ra số nhỏ nhất trong bảng chứa data số đó sẽ được thay đổi vị trí đầu tiên của trên bảng chứa data . Và quá trình này được lập lại trên các số thứ 2 và được xếp sau số đã được sắp xếp trên bảng . Quá trình này chỉ dừng lại khi bảng chứa data còn tồn tại 1 số còn lại chưa được sắp xếp và sau đó máy sẽ in ra 1 bảng chứa data đã được sắp xếp theo trình tự nhỏ nhất đến lớn nhất

Video chỉ cách hoạt động của seletion sort: <https://www.youtube.com/watch?v=xWBP4lzkoyM&t=14s>

Ở trong time complexity, Selection sort ở trong mọi case đều có giá trị O(n^2) và giá trị của O(n^2) độc lập với các giá trị data trong bảng chứa data . VÌ các ở mọi case trong selection sort đều là O(n^2) nên khi sử dụng cho các bảng chứa data chứa số lượng data lớn.

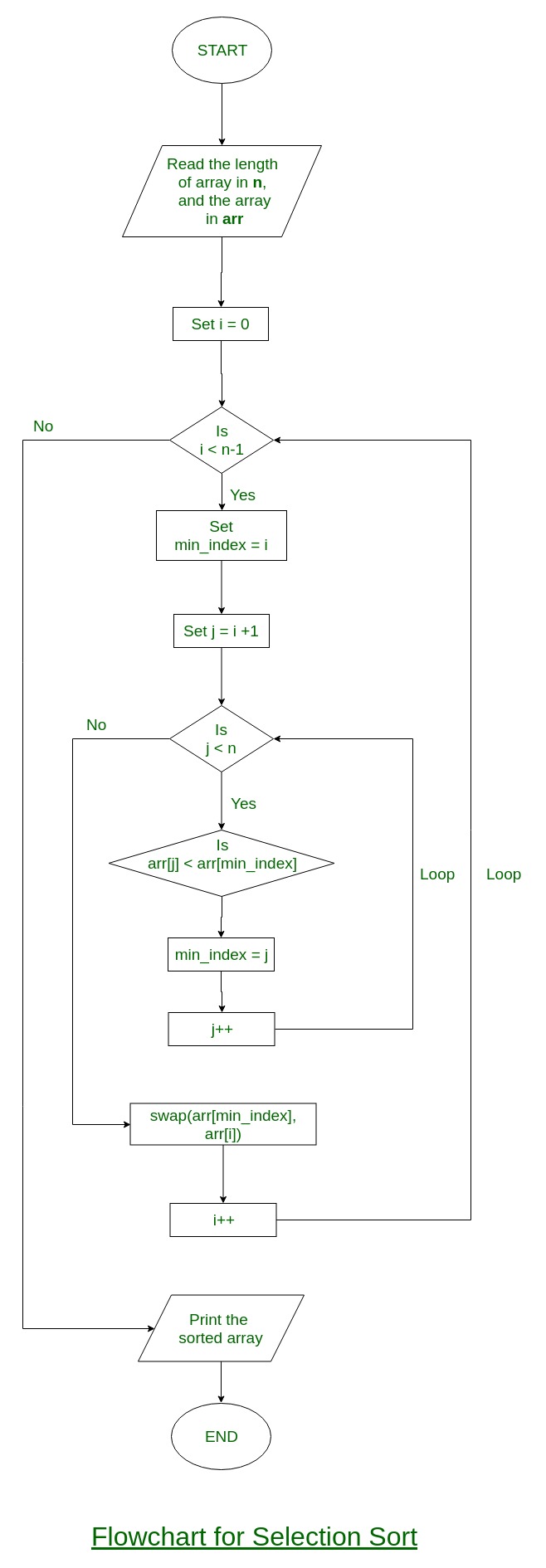
| Sorting Algorithm | Time Complexity | | | Space Complexity |
| --- | --- | --- | --- | --- |
|  | Best Case | Average Case | Worst Case | Worst Case |
| **Bubble Sort** | **Ω(N)** | **Θ(N2)** | **O(N2)** | **O(1)** |
| **Selection Sort** | **Ω(N2)** | **Θ(N2)** | **O(N2)** | **O(1)** |
| **Insertion Sort** | **Ω(N)** | **Θ(N2)** | **O(N2)** | **O(1)** |
| **Merge Sort** | **Ω(N log N)** | **Θ(N log N)** | **O(N log N)** | **O(N)** |
| **Heap Sort** | **Ω(N log N)** | **Θ(N log N)** | **O(N log N)** | **O(1)** |
| **Quick Sort** | **Ω(N log N)** | **Θ(N log N)** | **O(N2)** | **O(N log N)** |
| **Radix Sort** | **Ω(N k)** | **Θ(N k)** | **O(N k)** | **O(N + k)** |
| **Count Sort** | **Ω(N + k)** | **Θ(N + k)** | **O(N + k)** | **O(k)** |
| **Bucket Sort** | **Ω(N + k)** | **Θ(N + k)** | **O(N2)** | **O(N)** |

Application

Rất tốt khi việc viết các bộ nhớ là 1 quá trình rất tốn thời gian

Sử lý nhanh trong việc in và sắp xếp các chữ số

Rất quan trọng khi mà việc viết tốn thời gian hơn đọc trong bộ nhớ ở trong các EEPROM hay là Flash memory khi mà việc viết bằng thời gian hoạt động của cả bộ nhớ



Demo select sort using C (tăng dần)

|  |
| --- |
| #include **<**stdio.h**>**    void swap**(int** **\***xp**,** **int** **\***yp**)**  {  **int** temp **=** **\***xp;  **\***xp **=** **\***yp;  **\***yp **=** temp;  }  void selectionSort**(int** arr**[],** **int** n**)**  {  **int** i**,** j**,** min\_idx;    **/\*** One by one move boundary of unsorted subarray \*/  for **(**i **=** 0; i < n-1; i++)  {  **/\*** Find the minimum element **in** unsorted array \*/  min\_idx **=** i;  for **(**j **=** i**+**1; j < n; j++)  if **(**arr**[**j**]** **<** arr**[**min\_idx**])**  min\_idx **=** j;    **/\*** Swap the found minimum element with the first element\*/  swap**(&**arr**[**min\_idx**],** **&**arr**[**i**])**;  }  }    **/\*** Make an array **\*/**  void printArray**(int** arr**[],** **int** size**)**  {  **int** i;  for **(**i**=**0; i < size; i++)  printf**(**"%d "**,** arr**[**i**])**;  printf**(**"\n"**)**;  }  **/\*** **Test** case **\*/**  **int** main**()**  {  **int** arr**[]** **=** {71**,** 5**,** 42**,** 92**,** 11}; /\*Change the array here !!\*/  **int** n **=** sizeof**(**arr**)/**sizeof**(**arr**[**0**])**;  selectionSort**(**arr**,** n**)**;  printf**(**"Array = \n"**)**;  printArray**(**arr**,** n**)**;  return 0;  } |
|  |

Reverse version(giảm dần)

|  |
| --- |
| #include **<**stdio.h**>**  **int** main**()**{  **int** i**,** j**,** temp**,** arr**[**5**]=**{61**,**70**,**12**,**1**,**4};  **/\*** Logic of selection sort algorithm **\*/**  for**(**i**=**0;i<5;i++){  for**(**j**=**i**+**1;j<5;j++){  if**(**arr**[**i**]<** arr**[**j**])**{  temp**=**arr**[**i**]**;  arr**[**i**]=**arr**[**j**]**;  arr**[**j**]=**temp;  }  }  }  printf**(**"Sorted elements: "**)**;  for**(**i**=**0;i<5;i++)  printf**(**" %d"**,**arr**[**i**])**;  return 0;  } |