Thuật toán sắp xếp nhanh (Quick Sort)

Thuật toán Quick Sort được phát triển bởi **C.A.R**

Đúng như tên gọi, thuật toán sắp xếp nhanh là một thuật toán cho kết qua nhanh, gọn, nhẹ. Thuật toán này dựa trên việc chia một mảng thành các mảng nhỏ hơn.

Nếu so với các thuật toán sắp xếp khác như Insertion Sort hay sắp xếp nổi bọt (Bubble Sort), thì thuật toán sắp xếp nhanh cho tốc độ nhanh hơn đáng kể.

Thuật toán Quick sort là một thuật toán chia để trị (divide and Conquer Algorithm). Nó sẽ chọn một phần tử trong mảng làm điểm đánh dấu (**pivot**). Sau khi lựa chọn được điểm **pivot**, bước tiếp theo sẽ chia mảng thành nhiều mảng con dựa vào pivot đã chọn. Và lặp đi lặp lại như vậy cho đến khi kết thúc.

Tốc độ của thuật toán bị ảnh hưởng bởi việc chọn pivot. Có nhiều cách chọn pivot, dưới đây là một số cách:

- Luôn chọn phần tử đầu tiên của mảng làm pivot.
- Luôn chọn phần tử cuối cùng của mảng.
- Chọn ngẫu nhiên 1 phần tử trong mảng.
- Chọn phần tử có giá trị nằm giữa mảng (median element).

Để mình họa cho thuật toán sắp xếp nhanh, chúng ta cùng thực hành một bài toán: Sắp xếp mảng sau theo thứ tự tăng dần: [10, 7, 8, 9, 1, 5]

```
#include<iostream>
#include<cstdlib>

using namespace std;

// Swapping two values.
void swap(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
```

```
    *b = temp;
}

// Partitioning the array on the basis of values at high as pivot value.
int Partition(int a[], int low, int high)
{
    int pivot, index, i;
    index = low;
    pivot = high;

    // Getting index of the pivot.
    for(i=low; i < high; i++)
    {
        if(a[i] < a[pivot])
        {
            swap(&a[i], &a[index]);
            index++;
        }
    }
    // Swapping value at high and at the index obtained.
    swap(&a[pivot], &a[index]);

    return index;
}

// Random selection of pivot.
int RandomPivotPartition(int a[], int low, int high)
{
    int pvt, n, temp;
    n = rand();
    // Randomizing the pivot value in the given subpart of array.
    pvt = low + n%(high-low+1);

    // Swapping pivot value from high, so pivot value will be taken as a pivot while
partitioning.
    swap(&a[high], &a[pvt]);

    return Partition(a, low, high);
}
```

```
int QuickSort(int a[], int low, int high)
{
    int pindex;
    if(low < high)
    {
        // Partitioning array using randomized pivot.
        pindex = RandomPivotPartition(a, low, high);
        // Recursively implementing QuickSort.
        QuickSort(a, low, pindex-1);
        QuickSort(a, pindex+1, high);
    }
    return 0;
}

int main()
{
    int n, i;
    cout<<"\nEnter the number of data elements to be sorted: ";
    cin>>n;

    int arr[n];
    for(i = 0; i < n; i++)
    {
        cout<<"Enter element "<<i+1<<": ";
        cin>>arr[i];
    }

    QuickSort(arr, 0, n-1);

    // Printing the sorted data.
    cout<<"\nSorted Data ";
    for (i = 0; i < n; i++)
            cout<<"->"<<arr[i];

    return 0;
}
```

Revision #1
Created 6 May 2023 03:59:40 by Laptrinh.vn
Updated 27 June 2023 09:19:20 by Laptrinh.vn