**CHƯƠNG 3**

**Câu 1:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Trường hợp** | **Số lần so sánh** | **Số lần hoán vị** |
| Selection Sort | Tốt nhất |  | 0 |
| Xấu nhất |  |  |
| Insertion Sort | Tốt nhất | n-1 | 2(n-1) |
| Xấu nhất |  |  |
| Bubble Sort | Tốt nhất |  | 0 |
| Xấu nhất |  |  |
| Interchange Sort | Tốt nhất |  | 0 |
| Xấu nhất |  |  |
| Quick Sort | Tốt nhất |  | |
| Xấu nhất |  | |
| Heap Sort | Tốt nhất |  | |
| Xấu nhất |

Theo bảng độ phức tạp trên:

* Quick Sort là tối ưu nhất
* SelectionSort là kém tối ưu nhất

**Câu 2:**

Trong trường hợp duyệt danh sách có thứ tự, thì cả hai phương pháp đều như nhau. Vì trong trường hợp này, số lần so sánh của 2 phương pháp là như nhau.

**Câu 3**

Phương pháp khác là Merge Sort, là thuật toán có độ phức tạp trung bình và cũng sử dụng phương pháp chia để trị như Quick Sort. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp.

Độ  phức tạp thuật toán

* Trường hợp tốt: O(nlog(n))
* Trung bình: O(nlog(n))
* Trường hợp xấu: O(nlog(n))

Không gian bộ nhớ sử dụng: O(n)

Code:

void merge(int arr[], int l, int m, int r)

{

    int i, j, k;

    int n1 = m - l + 1;

    int n2 =  r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)

        L[i] = arr[l + i];

    for (j = 0; j < n2; j++)

        R[j] = arr[m + 1+ j];

    i = 0; // Khởi tạo chỉ số bắt đầu của mảng con đầu tiên

    j = 0; // Khởi tạo chỉ số bắt đầu của mảng con thứ hai

    k = l; // IKhởi tạo chỉ số bắt đầu của mảng lưu kết quả

    while (i < n1 && j < n2)

    {

        if (L[i] <= R[j])

        {

            arr[k] = L[i];

            i++;

        }

        else

        {

            arr[k] = R[j];

            j++;

        }

        k++;

    }

    while (i < n1)

    {

        arr[k] = L[i];

        i++;

        k++;

    }

    while (j < n2)

    {

        arr[k] = R[j];

        j++;

        k++;

    }

}

void mergeSort(int arr[], int l, int r)

{

    if (l < r)

    {

        int m = l+(r-l)/2;

        mergeSort(arr, l, m);

        mergeSort(arr, m+1, r);

        merge(arr, l, m, r);

    }

}