ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN – ĐIỆN TỬ

BÁO CÁO
CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Đề tài: Hybrid QuickSort Algorithm

Lớp: Hệ thống nhúng thông minh và IOT K67
Mã lớp học: 152455

Sinh viên thực hiện: Chu Thiên Phú 20224450

Giảng viên hướng dẫn: Tạ Thị Kim Huệ

Hà Nội, 1 - 2025

Sơ lược về Hybrid QuickSort Algorithm

1. **Định nghĩa về Hybri Algorithm**
   - Hybrid Algorithm là thuật toán kết hợp giữa 2 hoặc nhiều thuật toán dựa trên dữ liệu dầu vào.
   - Mục đích: tận dụng ưu điểm riêng của từng thuật toán để đạt hiệu quả tổng thể tốt hơn
2. **QuickSort và hybrid với InsertionSort**
   - QuickSort là một trong những thuật toán sắp xếp nhanh cho lượng dữ liệu lớn, nhanh hơn MergeSort và HeapSort từ 2 đến 3 lần nếu được triển khai tốt
   - Cơ chế kết hợp
     - Nếu kích thước của mảng nhỏ hơn một ngưỡng $\boldsymbol{k}$:
       - Tạm dừng quá trình phân chia của QuickSort.
       - Áp dụng Insertion Sort để sắp xếp các mảng con.
     - Sau khi xử lý xong toàn bộ mảng, mỗi phần tử cách vị trí cuối cùng nó không quá k vị trí. Việc sắp xếp lại với InsertionSort lúc này chỉ tốn $O(k \cdot n)$, trong đó k là hằng số.
3. **Tối ưu hoá bằng Tail Recursion:**
   - Tail Recursion giúp giảm mức sử dụng bộ nhớ khi triển khai đệ quy bằng cách:
     - Luôn xử lý phần mảng nhỏ hơn trước để tối thiểu độ sâu của cây đệ quy.
     - Chuyển phần đệ quy còn lại thằng tail call, từ đó tránh được việc sử dụng them khung ngăn xếp
4. **Trường hợp dữ liệu có nhiều phần tử trùng lặp:**
   - Khi dự liệu chứa nhiều phần tử giống nhau, QuickSort có thể hoạt động kém, ngay cả khi đã tối ưu hoá.
   - Giải pháp:
     - Sử dụng chiến lược phân chia tuyến tính (Dutch National Flag Partitioning) để chia mảng thành 3 nhóm: Nhỏ hơn, bằng và lớn hơn pivot.

- Phương pháp này giúp cải thiện hiệu suất với dữ liệu có nhiều phần tử trùng lặp.

5. Lợi ích của Hybrid QuickSort

- Kết hợp tính hiệu quả của QuickSort và sự đơn giản của InsertionSort cho mảng nhỏ
- Giảm đệ quy nhờ Tail Recursion.
- Dễ dàng mở rộng hoặc cải tiến với các chiến lược phân chia khác

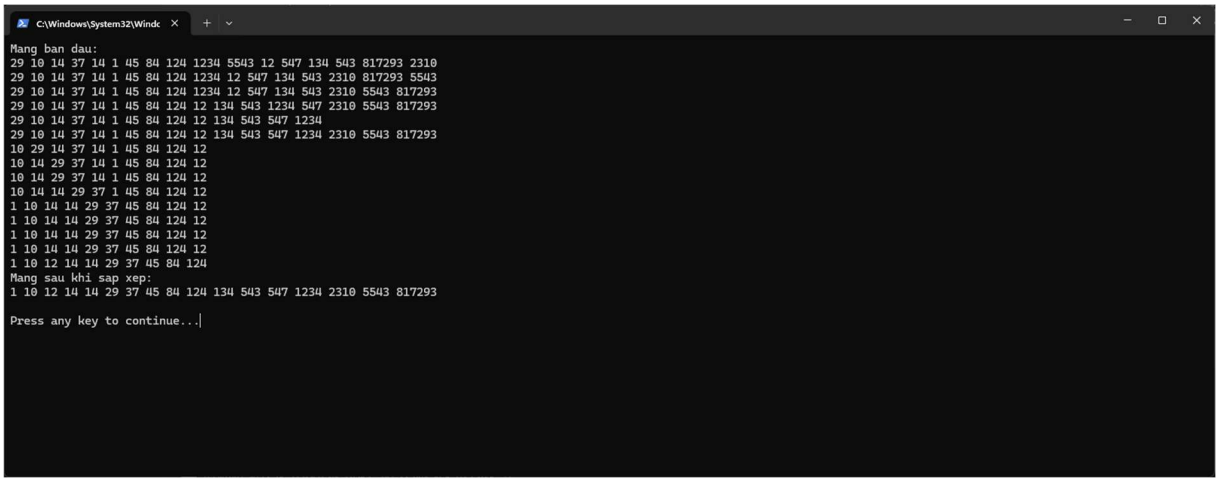
Mã giả

```
HybridQuickSort(A, low, high):
    while (low < high):
        if (high - low < k):                    Nếu high – low < k
            InsertionSort(A, low, high)         Dùng InsertionSort
            break
        else:                                   Nếu không
            pivot = Partition(A, low, high)     Thì gọi Partion để chia mảng
            if (pivot - low < high - pivot):
                HybridQuickSort(A, low, pivot - 1)
                 low = pivot + 1
            else:
                HybridQuickSort(A, pivot + 1, high)
                high = pivot - 1
InsertionSort(A, low, high):
    for i = low + 1 to high:
        value = A[i]
        j = i - 1
        while (j >= low and A[j] > value):
            A[j + 1] = A[j]
            j--
        A[j + 1] = value
```

Cài đặt chương trình bằng C

```c
#include <stdio.h>
#define K 10

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void insertionSort(int arr[], int low, int high) {
    for (int i = low + 1; i <= high; i++) {
        int value = arr[i];
        int j = i - 1;
        while (j >= low && arr[j] > value) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = value;
        printArray(arr, high + 1);
    }
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int pIndex = low;
    for (int i = low; i < high; i++) {
        if (arr[i] <= pivot) {
            int temp = arr[i];
            arr[i] = arr[pIndex];
            arr[pIndex] = temp;
            pIndex++;
        }
    }
    int temp = arr[pIndex];
    arr[pIndex] = arr[high];
    arr[high] = temp;
    return pIndex;
}

void hybridQuickSort(int arr[], int low, int high, int size) {
    while (low < high) {
        if (high - low < K) {
            insertionSort(arr, low, high);
            break;
        } else {
            int pivot = partition(arr, low, high);
            printArray(arr, size);
            if (pivot - low < high - pivot) {
                hybridQuickSort(arr, low, pivot - 1, size);
                low = pivot + 1;
            } else {
                hybridQuickSort(arr, pivot + 1, high, size);
                high = pivot - 1;
            }
        }
    }
}

int main() {
    int arr[] = {29, 10, 14, 37, 14};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Mang ban dau: \n");
    printArray(arr, n);

    hybridQuickSort(arr, 0, n - 1, n);

    printf("Mang sau khi sap xep: \n");
    printArray(arr, n);

    return 0;
}
```

Đầu ra


C:\Windows\System32\Windc
Mang ban dau:
29 10 14 37 14 1 45 84 124 1234 5543 12 547 134 543 817293 2310
29 10 14 37 14 1 45 84 124 1234 12 547 134 543 2310 817293 5543
29 10 14 37 14 1 45 84 124 1234 12 547 134 543 2310 5543 817293
29 10 14 37 14 1 45 84 124 12 134 543 1234 547 2310 5543 817293
29 10 14 37 14 1 45 84 124 12 134 543 547 1234
29 10 14 37 14 1 45 84 124 12 134 543 547 1234 2310 5543 817293
10 29 14 37 14 1 45 84 124 12
10 14 29 37 14 1 45 84 124 12
10 14 29 37 14 1 45 84 124 12
10 14 14 29 37 1 45 84 124 12
1 10 14 14 29 37 45 84 124 12
1 10 14 14 29 37 45 84 124 12
1 10 14 14 29 37 45 84 124 12
1 10 14 14 29 37 45 84 124 12
1 10 12 14 14 29 37 45 84 124
Mang sau khi sap xep:
1 10 12 14 14 29 37 45 84 124 134 543 547 1234 2310 5543 817293
Press any key to continue...

Tham khảo

Techie Delight: Hybrid QuickSort Algorithm | Techie Delight