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

1. **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:
       - 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 vị trí. Việc sắp xếp lại với InsertionSort lúc này chỉ tốn , trong đó là hằng số.
2. **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
3. **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.
4. **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**

A screen shot of a computer screen

Description automatically generated

**Đầu ra**

**A screenshot of a computer

Description automatically generated**

**Tham khảo**

Techie Delight: [Hybrid QuickSort Algorithm | Techie Delight](https://www.techiedelight.com/hybrid-quicksort/)