Các câu hỏi về báo cáo: Divide and Conquer

1. Giải thích rõ các kí hiệu trong công thức Recurrent relation?

$$T(n) = a * T\left(\frac{n}{b}\right) + f(n)$$

n là kích thước bài toán gốc
a là số bài toán con sau khi chia
n/b là kích thước bài toán con
f là chia phí cho việc chia và trị

2. Theo Recurrent relation thì độ phức tạp của Closest pair of points là bao nhiêu?

Giả sử sử dụng thuật toán sắp xếp O(nlogn)
Để xây dựng và tìm kiếm trong strip cần O(n)
Thì ta có

$T(n) = 2T(n/2) + O(n) + O(nLogn) + O(n)$

$T(n) = 2T(n/2) + O(nLogn)$

$T(n) = T(n \times Logn \times Logn)$

3. Ưu điểm "Giải quyết các vấn đề với độ phức tạp không quá lớn" là như thế nào?

Các bài toán thường được xem xét giải theo phương pháp Brute-Force đầu tiên và thường có độ phức tạp lớn.
Nếu có thể giải quyết bằng phương pháp Divide and Conquer thì độ phức tạp giảm đáng kể với không gian bài toán lớn.

4. Các thuật toán Divide and Conquer hoạt động như thế nào trên các hệ thống song song?

Các hệ thống song song có nhiều vi xử lý hoạt động đồng thời và sự trao đổi dữ liệu giữa chúng không nhất thiết phải được lên kế

hoạch cụ thể. Vì thế rất phù hợp với tính chất không gối lên nhau của các bài toán con.

5. Các thuật toán D&C tận dụng bộ nhớ đệm như thế nào?

Thường các bài toán có kích thước lớn chỉ được thực thi ở bộ nhớ trong. Tuy nhiên tốc độ truy cập dữ liệu vào bộ nhớ trong tốn rất nhiều thời gian so với bộ nhớ đệm. Nếu ta có thể chia nhỏ bài toán sao cho có thể thực thi trên bộ nhớ đệm thì có thể khắc phục điểm yếu này.

6. Nếu cài đặt đệ quy khiến chương trình thực thi chậm thì tại sao không sử dụng cách cài đặt khác?

Đúng là có cách để khắc phục điều này bằng các phương pháp khử đệ quy. Tuy nhiên khi tiếp cận một vấn đề, cách đệ quy sẽ thường được xem xét đầu tiên trong khi cách khử đệ quy có thể tốn chi phí để tìm ra.

7. Tại sao lại khó khăn trong việc tìm điều kiện dừng?

Chọn điều kiện dừng đơn giản sẽ giúp chương trình dễ hiểu hơn, ví dụ quicksort dừng khi input là một danh sách rỗng và không cần xử lý gì cả.
Vậy nếu sắp xếp danh sách n phần tử, trước khi dừng, thuật toán sẽ phải thực thi n lần gọi hàm đệ quy mà không có xử lý, chỉ trả về ngay lập tức.
Điều này gây nên sự lãng phí.
Mặt khác, nếu chọn điều kiện dừng lớn sẽ khiến chương trình phức tạp và tốn thời gian kiểm tra hơn rất nhiều.