UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

DIVIDE AND CONQUER

PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

CS112.L11.KHTN

Thông tin nhóm

1. Nguyễn Phú Quốc

18520343 - KHTN2018

2. Trần Trung Anh

18520473 - KHTN2018

3. Nguyễn Văn Tiến

18521489 - KHTN2018

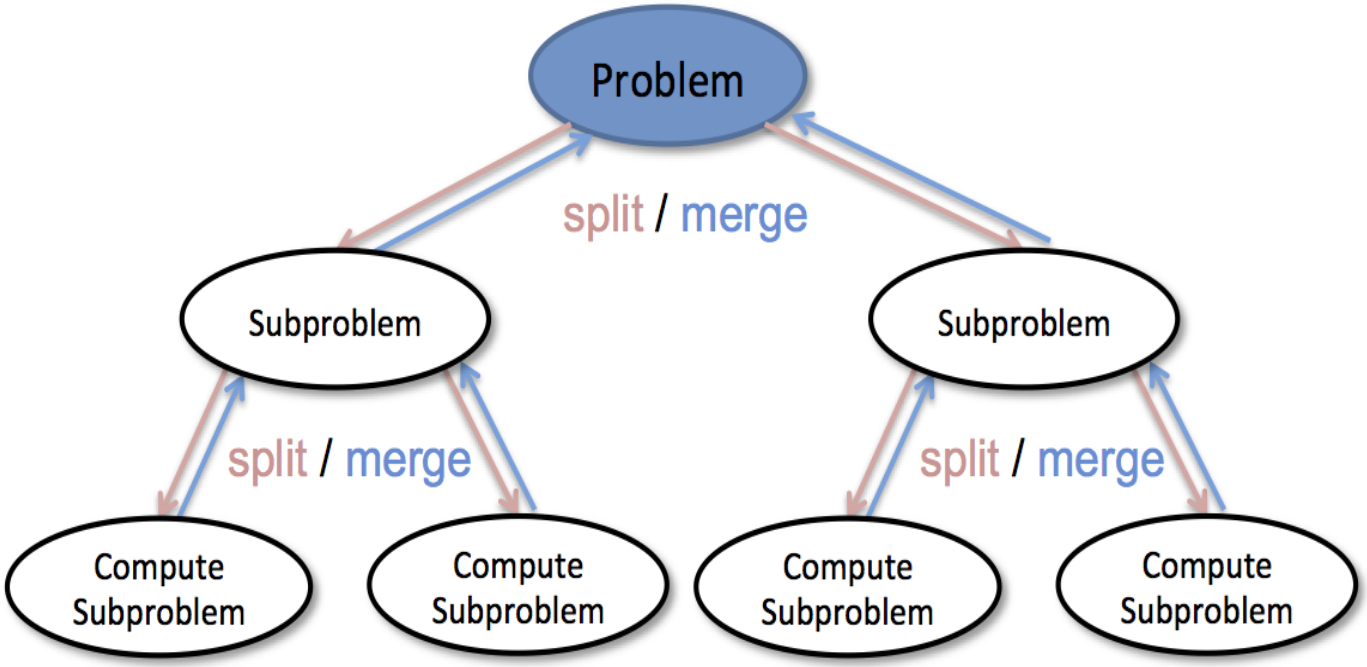
https://github.com/tiennvuit/CS112.L11.KHTN

Nội dung

1. Giới thiệu phương pháp thiết kế thuật toán: Chia và Trị.
2. Các bài toán điển hình.
3. Ưu và nhược điểm của phương pháp.
4. Tài liệu tham khảo.
5. Bài tập

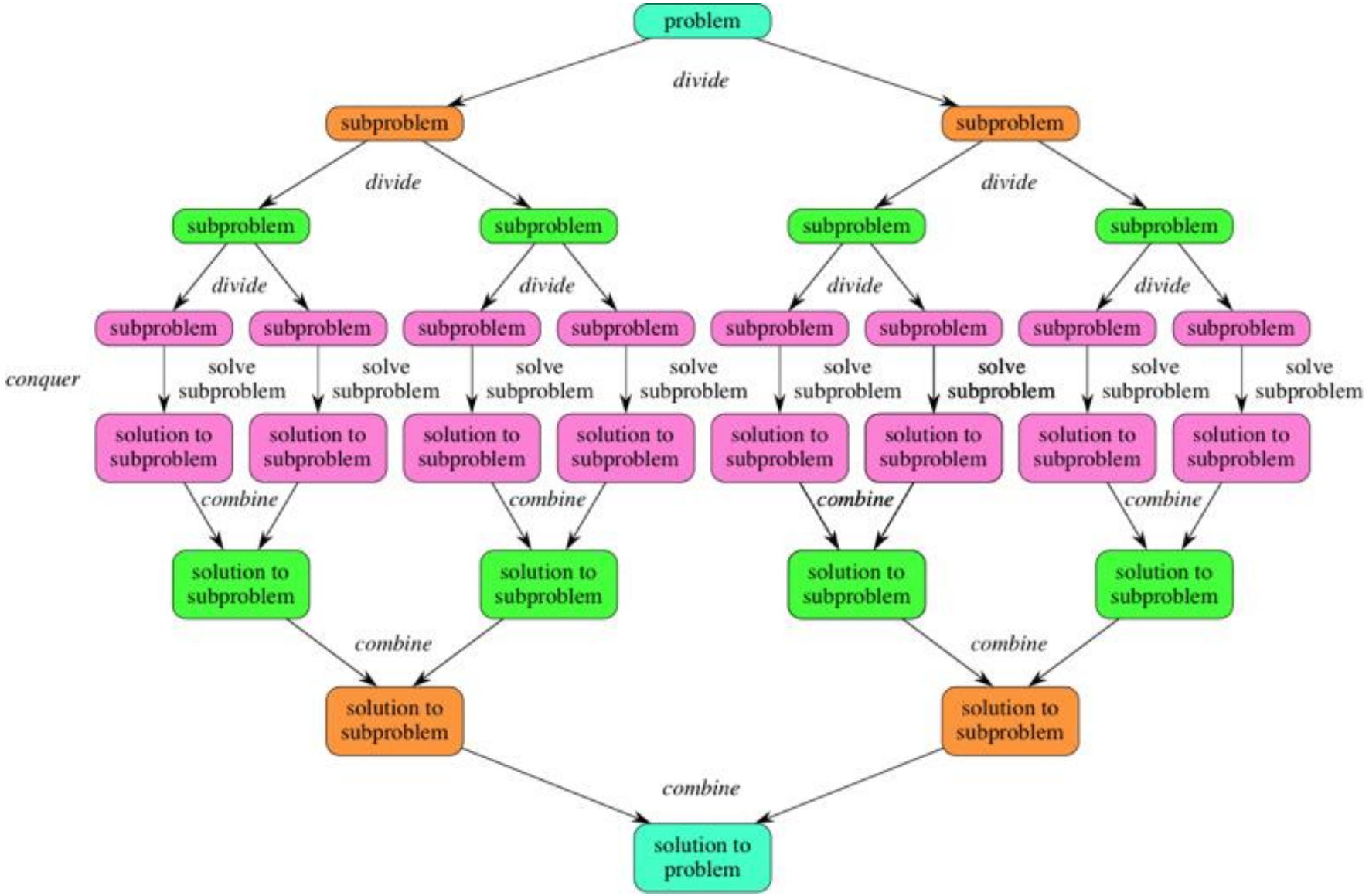
1. Giới thiệu

Chia để trị là 1 **phương pháp** áp dụng cho các bài toán có thể giải quyết bằng cách chia nhỏ ra thành các bài toán con từ việc giải quyết các bài toán con này. Sau đó lời giải của các bài toán nhỏ được tổng hợp lại thành lời giải cho bài toán ban đầu.



https://en.wikipedia.org/wiki/Divide-and-conquer_algorithm

1. Giới thiệu


problem
divide
subproblem
subproblem
divide
divide
subproblem
subproblem
subproblem
subproblem
divide
divide
divide
divide
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
conquer
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
combine
combine
combine
combine
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
combine
combine
solution to subproblem
solution to subproblem
combine
solution to problem

1. Giới thiệu

Bước 1: Chia/Tách nhỏ

Bài toán ban đầu sẽ được chia thành các bài toán con KHÔNG CÓ PHẦN CHUNG cho đến khi không thể chia nhỏ được nữa.

Bước 2: Trị/Giải quyết bài toán con

Tìm phương án để giải quyết cho bài toán con một cách cụ thể.

Bước 3: Kết hợp lời giải để suy ra lời giải

Khi đã giải quyết xong các bài toán nhỏ, lặp lại các bước giải quyết đó và kết hợp lại những lời giải để suy ra kết quả cần tìm.

Pseudocode template

```
divide_conquer(input J)
{
   // Base Case
   if (size of input is small enough) {
         solve directly and  return
   }

   // Divide Step
   divide J into two or more parts J1, J2,...

   // Recursive Calls
   call divide_conquer(J1) to get a subsolution S1
   call divide_conquer(J2) to get a subsolution S2
   ...
   // Merge Step
   Merge the subsolutions S1, S2,...into a global solution S
   return S
}
```

https://www2.slideshare.net/amrinderarora/divide-and-conquer-part-1

Phân tích độ phức tạp

- Đặt thời gian thực thi là $T(n)$. Khi đó, ta biểu diễn bài toán lớn về các bài toán con bằng công thức:

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

a là số nhánh, mỗi nhánh có kích thước là $\frac{n}{b}$, $f(n)$ là chi phí cho thao tác chia và trị

- Ví dụ: $T(n) = 2T\left(\frac{n}{2}\right) + 1$

→ Bài toán lớn có thể biểu diễn thành 2 bài toán con

Mỗi bài toán con có kích thước bằng ½ kích thước của bài toán lớn

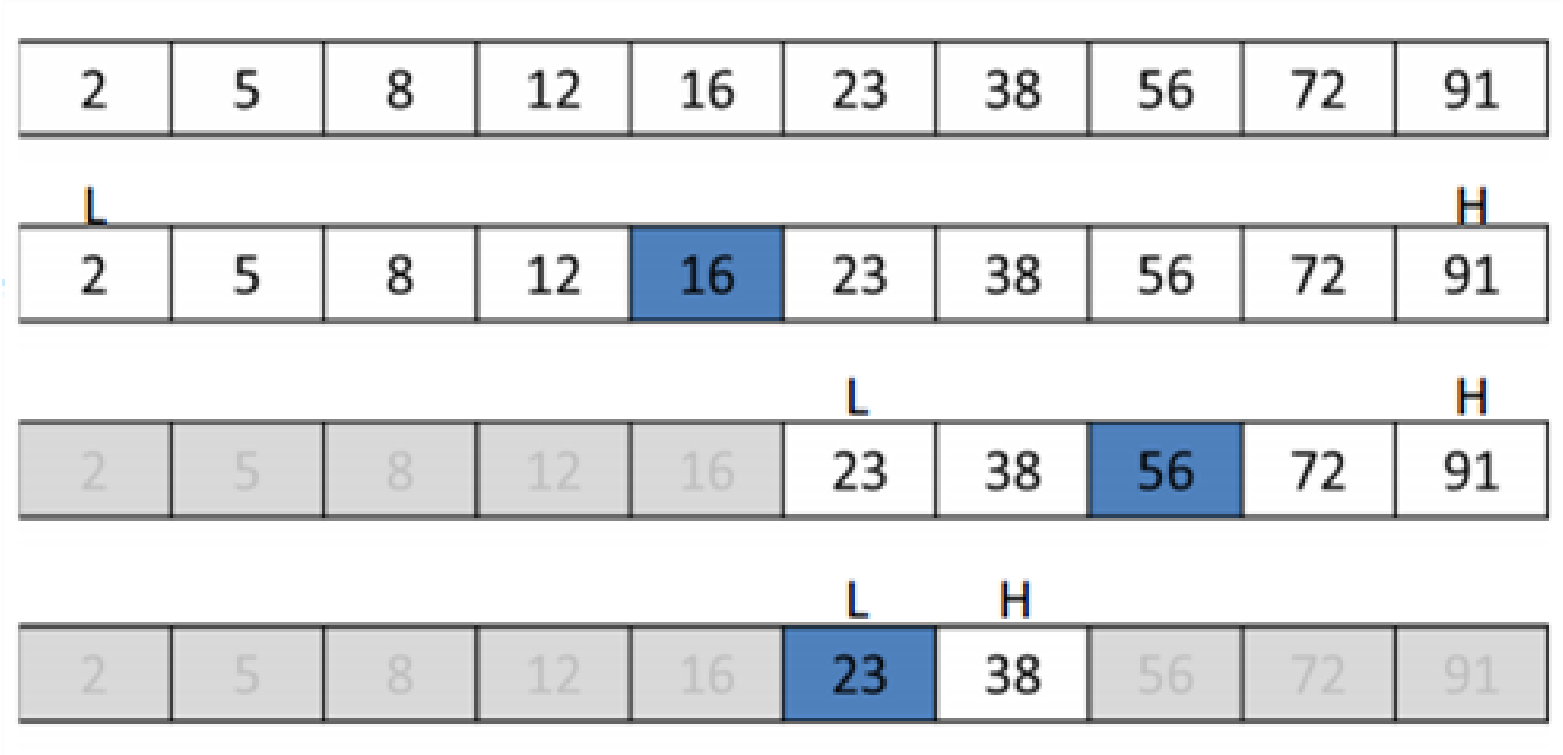
Chi phí cho thao tác chia và trị là 1

2. Các bài toán điển hình

- Binary Search
- Merge Sort
- Closest pair of points
- Karatsuba’s algorithm

2. Các bài toán điển hình

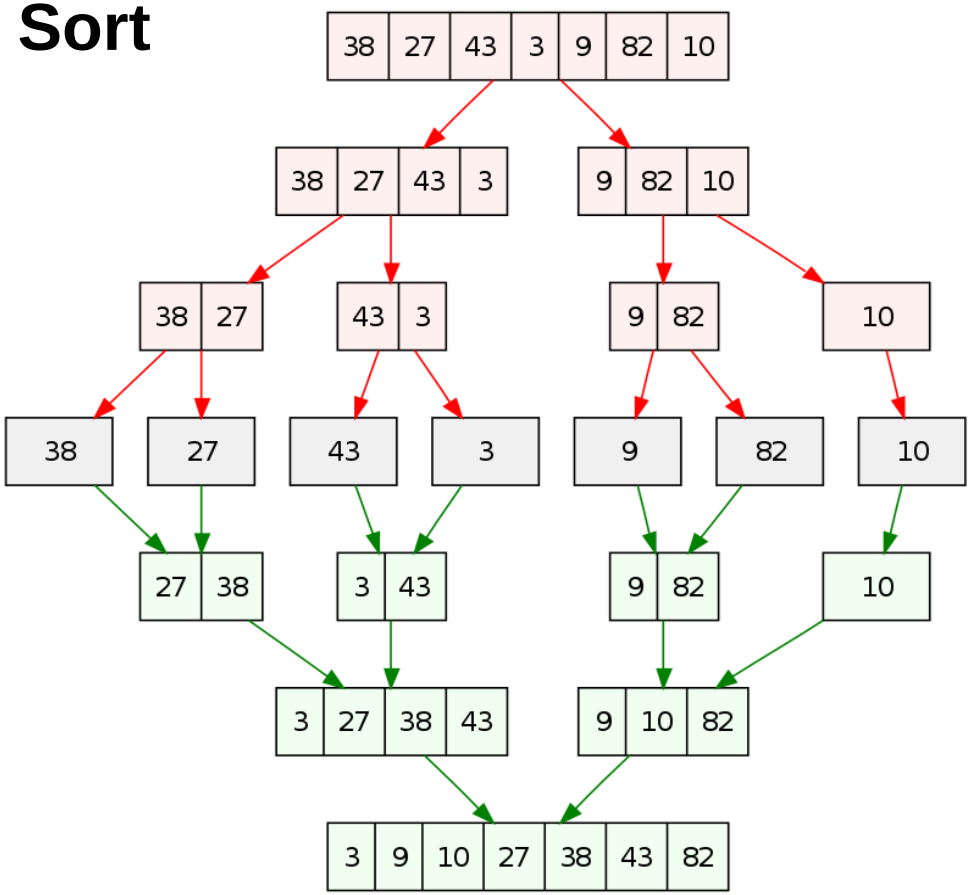
Binary Search (x = 23)



Độ phức tạp là O(log n)

2. Các bài toán điển hình

Merge Sort



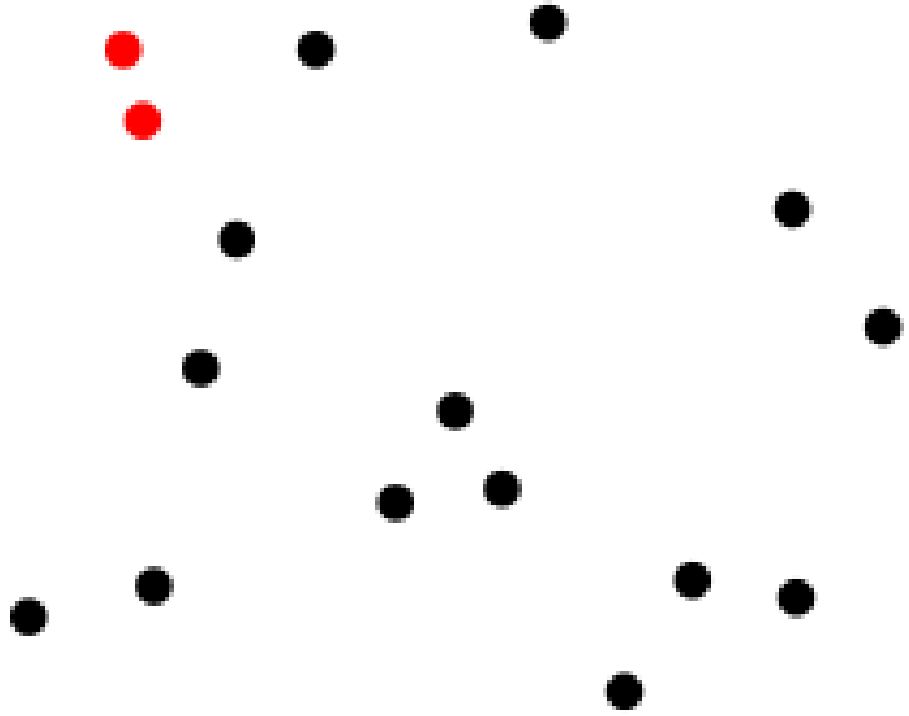
Độ phức tạp O(nlogn)

2. Các bài toán điển hình

Closest pair of points

Cho n điểm trên hệ trục tọa độ Oxy.

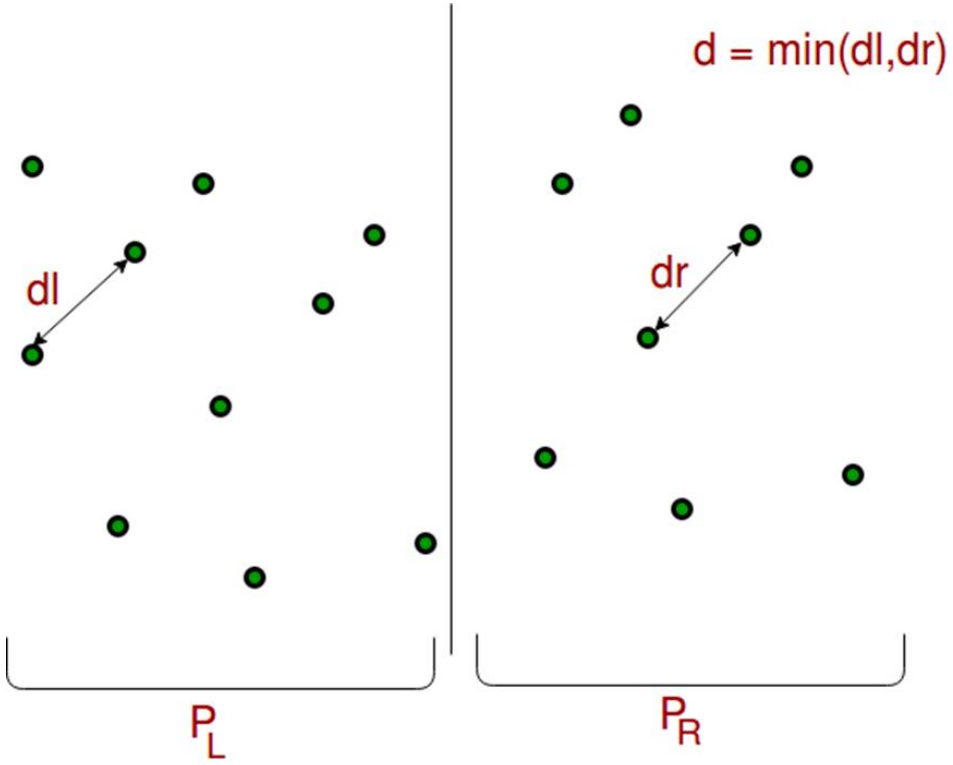
Tìm cặp điểm có khoảng cách ngắn nhất.



2. Các bài toán điển hình

Closest pair of points

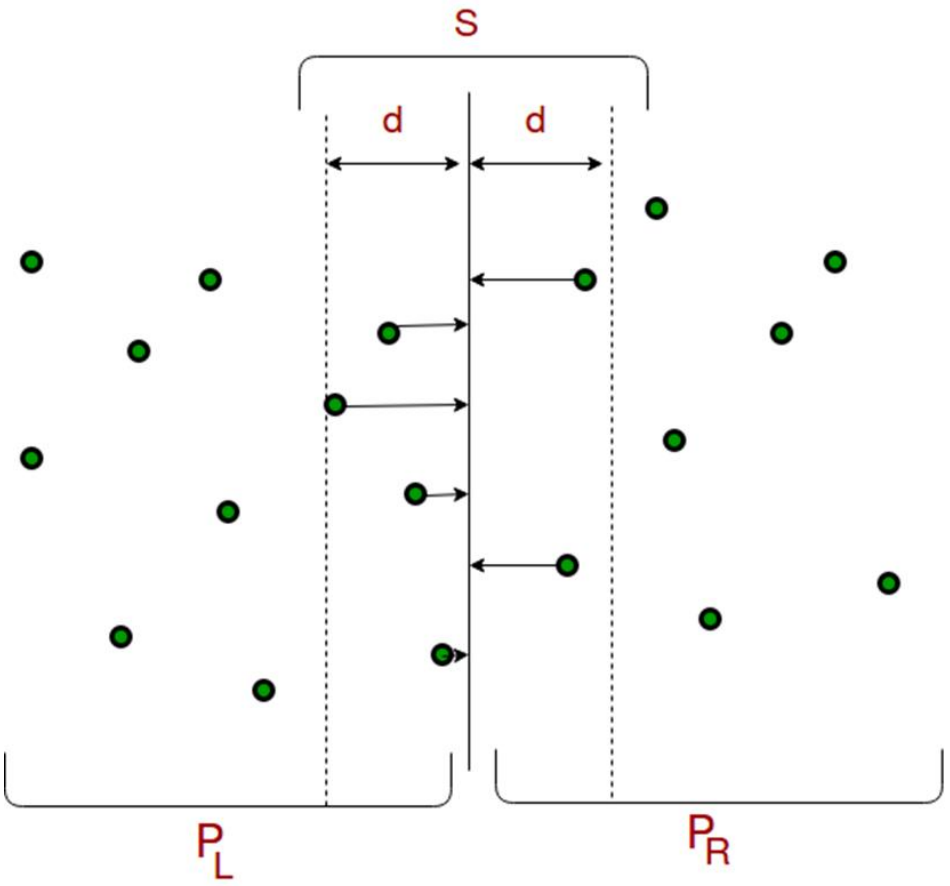
1. Sắp xếp các điểm theo trục x.
2. Đặt mid là vị trí là tọa độ điểm ở giữa.
3. Chia dãy được cho thành 2 phần tại mid.
4. Tìm khoảng cách nhỏ nhất ở 2 phần (dl, dr).
5. Đặt d = min(dl, dr).



2. Các bài toán điển hình

Closest pair of points

6. Đặt đường thẳng P tại mid.
7. Đặt strip[] gồm các điểm có khoảng cách (x, mid.x) nhỏ hơn d.
8. Sắp xếp strip theo y.
9. Tìm khoảng cách nhỏ nhất của các điểm trong strip.



2. Các bài toán điển hình

Closest pair of points

Độ phức tạp:

Sắp xếp: *O(nlogn)*

Xây dựng và tìm kiếm trong strip: *O(n)*

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

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

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

2. Các bài toán điển hình

Karatsuba's algorithm

Cho 2 số nguyên a và b. Trong đó a có n chữ số và b có m chữ số.

Tính độ phức tạp của khi tính tích a và b.

2. Các bài toán điển hình

Karatsuba's algorithm

Công thức nhân hai số lớn X và Y sử dụng ba phép nhân các số nhỏ hơn.

Cho x và y là hai số **n** bit trong hệ số B và $m < \boldsymbol{n}$, ta có

$$x = x_1 B^m + x_0$$

$$y = y_1 B^m + y_0$$

2. Các bài toán điển hình

Karatsuba's algorithm

$$xy = (x_1 B^m + x_0)(y_1 B^m + y_0)$$
$$= x_1 y_1 B^{2m} + x_1 y_0 B^m + x_0 y_1 B^m + x_0 y_0$$
$$= x_1 y_1 B^{2m} + (x_1 y_0 + x_0 y_1) B^m + x_0 y_0$$

Với $\begin{cases} z_2 = x_1 y_1 \\ z_1 = x_1 y_0 + x_0 y_1 \\ z_0 = x_0 y_0 \end{cases}$

Có 4 phép nhân cần thực hiện

2. Các bài toán điển hình

Karatsuba's algorithm

Nhận thấy

$$z_1 = (x_0 - x_1)(y_1 - y_0) + z_2 + z_0$$

Với $\begin{cases} z_2 = x_1 y_1 \\ z_1 = (x_0 - x_1)(y_1 - y_0) + z_2 + z_0 \\ z_0 = x_0 y_0 \end{cases}$

Số phép nhân giảm còn 3!

2. Các bài toán điển hình

Karatsuba's algorithm

Thực hiện phép nhân $12345 \cdot 6789$

12345 = **12** · *1000* + **345**

6789 = **6** · *1000* + **789**

z_2 = **12** × **6** = 72

z_0 = **345** × **789** = 272205

z_1 = (**12** + **345**) × (**6** + **789**) − z_2 − z_0 = 357 × 795 − 72 − 272205 = 283815 − 72 − 272205 = 11538

result = $z_2 \cdot (B^m)^2 + z_1 \cdot (B^m)^1 + z_0 \cdot (B^m)^0$, i.e.

result = 72 · 1000^2 + 11538 · *1000* + 272205 = **83810205**.

Độ phức tạp: $O(n^{\log_2 3})$

3. Ưu và nhược điểm của phương pháp

Ưu điểm

- Độ phức tạp thường thấp hơn cách tiếp cận Brute-Force.
- Thuận tiện khi thực thi trên các hệ thống song song.

  Các bài toán khi được chia nhỏ có thể được xử lý song song trên các luồng, tiến trình chạy song song.

- Tận dụng bộ nhớ đệm để tính toán.

  Các bài toán con có kích thước nhỏ hơn được thao tác trên bộ nhớ có tốc độ truy xuất nhanh(cache, thanh ghi) thay vì sử dụng các bộ nhớ lưu trữ RAM.

3. Ưu và nhược điểm của phương pháp

Nhược điểm

- Cài đặt đệ quy khiến chương trình thực thi chậm và dễ xảy ra lỗi.

  Giải pháp: sử dụng các cấu trúc dữ liệu (stack, queue) để khử đệ quy.

- Có thể cần thêm bộ nhớ để lưu trữ tạm các phần chia nhỏ.

- Khó khăn trong việc lựa chọn điều kiện dừng(lựa chọn kích thước của bài toán có giải giải ngay lập tức).

4. Khi nào dùng Chia và trị?

1. Chia bài toán lớn thành các bài toán con **không gối nhau** là các trường hợp nhỏ hơn của cùng một bài toán.

2. Mỗi bài toán con có thể được giải một cách đệ quy (mỗi trường hợp của bài toán con có bản chất giống hệt nhau).

3. Lời giải của mỗi bài toán con có thể được kết hợp để giải quyết bài toán lớn được phân rã.

5. Tài liệu tham khảo

- Cormen, Thomas H., et al. Introduction to algorithms. MIT press, 2009.
- Divide-and-conquer algorithms. Chapter 2. EECS-Berkeley. PDF
- Divide-and-conquer algorithm. Wikipedia
- Divide and Conquer Algorithm Meaning: Explained with Examples GeeksForGeeks
- Closest Pair of Points using Divide and Conquer algorithm GeeksForGeeks
- Karatsuba algorithm Wikipedia

THANKS FOR LISTENING