Bài tập
Phân tích thiết kế thuật toán

Giải các phương trình đệ quy với $T(1) = 1$ và

- **$T(n) = 3T(n/2) + n$**
- ***Cách 1***:
- Ta có: $a = 3$ và $b = 2$ và nghiệm thuần nhất là $n^{\log_b a} = n^{\log_2 3}$
- $d(n) = n \rightarrow d(b) = 2$. Vì $a = 3 > d(b) = 2$ nên nghiệm riêng cũng là nghiệm thuần nhất → $T(n) = O(n^{\log_2 3})$
- ***Cách 2***:
- Áp dụng định lý thợ: Ta có $\alpha = 1 < \log_b a = \log_2 3$ → rơi vào trường hợp 1 và khi đó $T(n) = O(n^{\log_b a}) = O(n^{\log_2 3})$

Giải các phương trình đệ quy với $T(1) = 1$ và

- **$T(n) = 3T(n/2) + n^2$**
- ***Cách 1***:
- Ta có: $a = 3$ và $b = 2$ và nghiệm thuần nhất là $n^{\log_b a} = n^{\log_2 3}$
- $d(n) = n^2 \rightarrow d(b) = 4$. Vì $a = 3 < d(b) = 4$ nên nghiệm riêng là $O(n^{\log_b d(b)}) = O(n^{\log_2 4}) \rightarrow T(n) = O(n^2)$
- ***Cách 2***:
- Áp dụng định lý thợ: Ta có $\alpha = 2 > \log_b a = \log_2 3 \rightarrow$ rơi vào trường hợp 3 và khi đó $T(n) = O(f(n)) = O(n^2)$

Giải các phương trình đệ quy với $T(1) = 1$ và

- **$T(n) = 8T(n/2) + n^3$**
- ***Cách 1***:
- Ta có: $a = 8$, $b = 2$ → nghiệm thuần nhất là $n^{\log_b a} = n^{\log_2 8} = n^3$
- $d(n) = n^3$ → $d(b) = 8$. Vì $a = 8 = d(b) = 8$ nên nghiệm riêng là $O(n^\alpha \log n) = O(n^3 \log n)$ → $T(n) = O(n^3 \log n)$
- ***Cách 2***:
- Áp dụng định lý thợ: Ta có $\alpha = 3 = \log_b a = \log_2 8$ → rơi vào trường hợp 2 và khi đó $T(n) = O(n^\alpha \log n) = O(n^3 \log n)$

Giải các phương trình đệ quy với $T(1) = 1$ và

- **$T(n) = 4T(n/3) + n$**
- ***Cách 1***:
- Ta có: $a = 4$, $b = 3$ → nghiệm thuần nhất là $n^{\log_b a} = n^{\log_3 4}$
- $d(n) = n$ → $d(b) = 3$. Vì $a = 4 > d(b) = 3$ nên nghiệm riêng cũng chính là nghiệm thuần nhất → $T(n) = O(n^{\log_3 4}) = O(n^{2 \log_3 2})$
- ***Cách 2***:
- Áp dụng định lý thợ: Ta có $\alpha = 1 < \log_b a = \log_3 4$ → rơi vào trường hợp 1 và khi đó $T(n) = O(n^{\log_b a}) = O(n^{2 \log_3 2})$

Giải các phương trình đệ quy với $T(1) = 1$ và

- **$T(n) = 4T(n/3) + n^2$**
- ***Cách 1***:
- Ta có: $a = 4$, $b = 3$ → nghiệm thuần nhất là $n^{\log_b a} = n^{\log_3 4}$
- $d(n) = n^2$ → $d(b) = 9$. Vì $a = 4 < d(b) = 9$ nên nghiệm riêng là $O(n^{\log_b d(b)}) = O(n^{\log_3 9})$ → $T(n) = O(n^2)$
- ***Cách 2***:
- Áp dụng định lý thợ: Ta có $\alpha = 2 > \log_b a = \log_3 4$ → rơi vào trường hợp 3 và khi đó $T(n) = O(f(n)) = O(n^2)$

Giải các phương trình đệ quy với $T(1) = 1$ và

- **$T(n) = 9T(n/3) + n^2$**
- ***Cách 1***:
- Ta có: $a = 9$, $b = 3$ → nghiệm thuần nhất là $n^{\log_b a} = n^{\log_3 9} = n^2$
- $d(n) = n^2$ → $d(b) = 9$. Vì $a = 9 = d(b) = 9$ nên nghiệm riêng là $O(n^\alpha \log n) = O(n^2 \log n)$→ $T(n) = O(n^2 \log n)$
- ***Cách 2***:
- Áp dụng định lý thợ: Ta có $\alpha = 2 = \log_b a = \log_3 9 = 2$ → rơi vào trường hợp 2 và khi đó $T(n) = O(n^\alpha \log n) = O(n^2 \log n)$

Giải các phương trình đệ quy với $T(1) = 1$ và

- **$T(n) = 2T(n/2) + \log n$**
- Ta có:

$$T(n) = \log_2 n + \underbrace{2\log_2(\frac{n}{2}) + 4\log_2(\frac{n}{4}) + 8\log_2(\frac{n}{8}) + \ldots}_{\log_2 n \text{ times}}$$

$$T(n) = \log_2 n + \sum_{k=1}^{\log_2 n} 2^k \log_2(\frac{n}{2^k})$$

Giải các phương trình đệ quy với $T(1) = 1$ và

- **$T(n) = 2T(n/2) + \log n$**
- Đặt $m = \log_2 n$, ta có:

$$\sum_{k=1}^{m} 2^k \log_2 \left(\frac{n}{2^k}\right) =$$

$$= \sum_{k=1}^{m} 2^k (\log_2 n - k) =$$

$$= \log_2 n \sum_{k=1}^{m} 2^k - \sum_{k=1}^{m} k 2^k =$$

$$= \log_2 n (2^{m+1} - 2) - (m 2^{m+1} - 2^{m+1} + 2)$$

Chú ý: $\sum_{k=1}^{n} k\, 2^k = 2\left(2^n n - 2^n + 1\right)$

Giải các phương trình đệ quy với $T(1) = 1$ và

- **$T(n) = 2T(n/2) + \log n$**
- Thay m bằng $\log_2 n$, ta có:

$$T(n) = \log_2 n + 2n \log_2 n - 2 \log_2 n - 2n \log_2 n + 2n - 2$$

$$= 2n - \log_2 n - 2 = \Theta(n)$$

Phương pháp phân tích thiết kế chia để trị

Bài 2: Tìm tổng lớn nhất của đoạn con liên tục

Bài toán

- Input: $a_1, a_2, \ldots, a_n$
- Output: $i, j \mid a_i + a_{i+1} + .. + a_j$ là lớn nhất

Phương pháp phân tích thiết kế chia để trị

Thuật toán:

Gọi f[i] là tổng i phần tử đầu tiên, f[i] được tính như sau:

- f[0] = 0
- f[i] = f[i-1] + a[i]

Tổng các phần tử từ i đến j là: f[j] – f[i-1]

Phương pháp phân tích thiết kế chia để trị

Ví dụ với mảng a[] = {1, 3, -5, 2, 7, 6, -2, 4, -3, 1}

- f[0] = 0, f[i] = f[i-1] + a[i]
- Tổng các phần tử từ i đến j là f[j] – f[i-1]

Ta có:

f[0] = 0, f[1] = 0 + 1 = 1, f[2] = 1 + 3 = 4, f[3] = 4 + (-5) = -1,

f[4] = -1 + 2 = 1, f[5] = 1 + 7 = 8, f[6] = 8 + 6 = 14,

f[7] = 14 + (-2) = 12, f[8] = 12 + 4 = 16, f[9] = 16 + (-3) = 13,

f[10] = 13 + 1 = 14

Kết quả: tổng đoạn con từ a[4] đến a[8] là 17

Phương pháp phân tích thiết kế chia để trị

Thuật toán:

```
int maxSubArraySum(int a[], int n)  {
    int f[n], max_sum = 0;
    f[0] = 0;
    for (int i = 1; i <= n; i++)
            f[i] = f[i-1] + a[i];

    for (int i = 1; i < n; i++)
            for (int j = i; j < n;j++)
                    max_sum = max(f[j] - f[i-1], max_sum);

    return max_sum;
}
```

Độ phức tạp: $T(n) = O(n^2)$

Phương pháp phân tích thiết kế chia để trị

Thuật toán: $T(n) = O(n)$

```
int maxSubArraySum(int a[], int n)
{
   int max_so_far = a[0];
   int curr_max = a[0];
   for (int i = 1; i < n; i++)
   {
        curr_max = max(a[i], curr_max + a[i]);
        max_so_far = max(max_so_far, curr_max);
   }
   return max_so_far;
}
```

Phương pháp phân tích thiết kế chia để trị

Bài 3: Sửa đổi bài 9 để tìm dãy con liên tục có tổng lớn nhất

Phương pháp phân tích thiết kế chia để trị

Bài 4: Phân tích thiết kế đánh giá độ phức tạp của thuật toán sắp xếp nhanh (Quick sort)

Phương pháp phân tích thiết kế chia để trị

Bài 4: Quick sort

- Là thuật toán sắp xếp dựa trên kỹ thuật chia để trị
- Các bước:
  - Cơ sở: dãy chỉ còn không quá 1 phần tử thì dãy đó đã được sắp xếp => không cần thao tác thêm gì cả
  - Chia: gồm hai công việc
    - Chọn một phần tử trong dãy làm phần tử chốt p (pivot)
    - Phân hoạch: chia dãy đã cho thành hai dãy con, dãy con trái (L) sẽ gồm những phần tử nhỏ hơn hoặc bằng phần tử chốt và dãy con phải (R) là những phần tử lớn hơn phần tử chốt
  - Trị: lặp lại một cách đệ quy thuật toán với 2 dãy con trái và phải
  - Tổng hợp: dãy được sắp xếp L,p,R

Phương pháp phân tích thiết kế chia để trị

Bài 4: Quick sort

Với thuật toán Quick Sort thì việc chọn phần tử chốt có vai trò quyết định đối với hiệu quả của thuật toán. Các cách chọn phần tử chốt:

- Chọn phần tử đầu tiên
- Chọn phần tử cuối cùng
- Chọn phần tử đúng giữa dãy
- Chọn phần tử trung vị trong 3 phần tử đứng đầu, đứng cuối và đứng giữa
- Chọn phần tử ngẫu nhiên

Phương pháp phân tích thiết kế chia để trị

Bài 4: Quick sort, chốt là phần tử cuối dãy

```
int partition (int arr[], int low, int high)  {
    int pivot = arr[high];    // pivot
    int i = (low - 1);        // Index of smaller element

    for (int j = low; j < high; j++)  {
        // If current element is smaller than the pivot
        if (arr[j] < pivot)  {
            i++;              // increment index of smaller element
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
```

Phương pháp phân tích thiết kế chia để trị

Bài 4: Quick sort, chốt là phần tử cuối dãy

Ví dụ: arr[] = {10, 80, 30, 90, 40, 50, 70}
Chỉ số: 0 1 2 3 4 5 6

low = 0, high = 6, pivot = arr[high] = 70
Khởi tạo chỉ mục của phần tử nhỏ hơn, i = -1

Di chuyển các phần tử từ j = low đến high - 1
j = 0: Vì arr[j] = 10 <= pivot = 70, thực hiện i++ và hoán đổi (arr[i], arr[j])
i = 0
arr[] = {10, 80, 30, 90, 40, 50, 70} // Không thay đổi vì i và j đều bằng 0

j = 1: Vì arr[j] = 80 > pivot = 70, không làm gì cả
// Không thay đổi trong i và arr[]

Phương pháp phân tích thiết kế chia để trị

Bài 4: Quick sort, chốt là phần tử cuối dãy

Ví dụ: arr [] = {10, 80, 30, 90, 40, 50, 70}
Chỉ số: 0 1 2 3 4 5 6

j = 2: Vì arr[j] = 30 <= pivot = 70, thực hiện i++ và hoán đổi (arr[i], arr[j])
i = 1
arr[] = {10, 30, 80, 90, 40, 50, 70} // Chúng ta hoán đổi 80 và 30

j = 3: Vì arr[j] = 90 > pivot = 70, không làm gì cả
// Không thay đổi trong i và arr[]

j = 4: Vì arr[j] = 40 <= pivot = 70, thực hiện i++ và hoán đổi (arr[i], arr[j])
i = 2
arr[] = {10, 30, 40, 90, 80, 50, 70} // 80 và 40 được hoán đổi

Phương pháp phân tích thiết kế chia để trị

Bài 4: Quick sort, chốt là phần tử cuối dãy

Ví dụ: arr [] = {10, 80, 30, 90, 40, 50, 70}
Chỉ số: 0 1 2 3 4 5 6

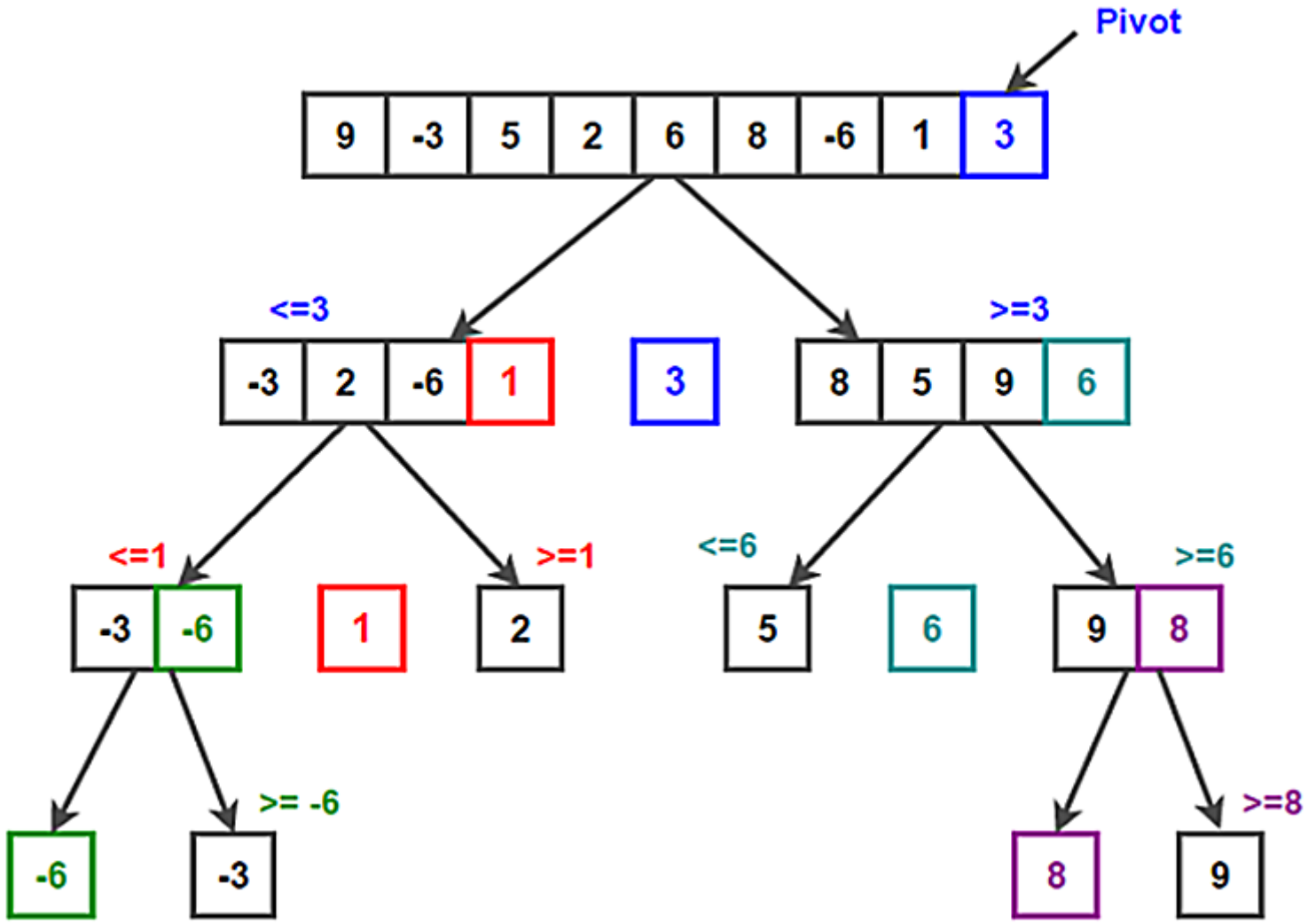
j = 5: Vì arr[j] = 50 <= pivot = 70, thực hiện i++ và hoán đổi arr[i] với arr[j]
i = 3
arr[] = {10, 30, 40, 50, 80, 90, 70} // 90 và 50 được hoán đổi

Chúng ta đi ra khỏi vòng lặp vì j bây giờ bằng high - 1.
Cuối cùng, đặt chốt ở vị trí chính xác bằng cách hoán đổi arr[i + 1] và arr[high] (là chốt)
arr[] = {10, 30, 40, 50, 70, 90, 80} // 80 và 70 được hoán đổi

Bây giờ 70 đã ở đúng vị trí của nó. Tất cả các phần tử nhỏ hơn 70 ở trước nó và tất cả các phần tử lớn hơn 70 ở sau nó.

Phương pháp phân tích thiết kế chia để trị

Bài 4: Quick sort, chốt là phần tử cuối dãy



Phương pháp phân tích thiết kế chia để trị

Bài 4: Độ phức tạp của thuật toán Quick sort

Độ phức tạp thuật toán phụ thuộc vào việc phần tử nào được chọn là phần tử chốt

- Phân hoạch không cân bằng: một bài toán con có kích thước n-1 và bài toán kia có kích thước là 0. Đó là trường hợp xấu nhất xảy ra khi dãy đã cho là dãy đã được sắp xếp và phần tử chốt được chọn là phần tử đầu của dãy => độ phức tạp thuật toán sẽ là $O(n^2)$
- Phân hoạch hoàn hảo: mỗi lần phân hoạch ta đều chọn được phần tử median (phần tử lớn hơn hay bằng nửa số phần tử và nhỏ hơn hay bằng nửa số phần tử còn lại) làm mốc. Khi đó dãy được phân hoạch thành hai phần bằng nhau, và ta cần $\log_2(n)$ lần phân hoạch thì sắp xếp xong. Ta cũng dễ nhận thấy trong mỗi lần phân hoạch ta cần duyệt qua n phần tử. Vậy độ phức tạp trong trường hợp tốt nhất thuộc $O(n\log_2(n))$

Nhân ma trận

$$\left.\begin{array}{l} Input: A=[a_{ij}], B=[b_{ij}] \\ Output: C=[c_{ij}]=A.B \end{array}\right\} \; i, j = 1,2,\ldots,n$$

$$\begin{bmatrix} c_{11} & c_{12}\ldots c_{1n} \\ c_{21} & c_{22}\ldots c_{2n} \\ \ldots & \\ c_{n1} & c_{n2}\ldots c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12}\ldots a_{1n} \\ a_{21} & a_{22}\ldots a_{2n} \\ \ldots & \\ a_{n1} & a_{n2}\ldots a_{nn} \end{bmatrix} . \begin{bmatrix} b_{11} & b_{12}\ldots b_{1n} \\ b_{21} & b_{22}\ldots b_{2n} \\ \ldots & \\ b_{n1} & b_{n2}\ldots b_{nn} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik}.b_{kj}$$

Thuật toán thông dụng

```
for i ← 1 to n do
    for j ← 1 to n do
        c_ij ← 0
        for k ← 1 to n do
            c_ij ← c_ij + a_ik . b_kj
```

Thuật toán thông dụng

for $i \leftarrow 1$ **to** n **do**

  for $j \leftarrow 1$ **to** n **do**

    $c_{ij} \leftarrow 0$

    for $k \leftarrow 1$ **to** n **do**

      $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot B_{kj}$

Thời gian thực hiện = $O(n^3)$

$n = 64 \rightarrow n^3 = 64*64*64 = 262144$ bước lặp

Thuật toán chia để trị

Ý tưởng: ma trận $n \times n = 2 \times 2$ ma con trận cấp $(n/2) \times (n/2)$:

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

$$C = A.B$$

$$\left.\begin{aligned} r &= ae + bg \\ s &= af + bh \end{aligned}\right\} \text{ 8 phép nhân ma trận con cấp } (n/2) \times (n/2)$$

$$\left.\begin{aligned} t &= ce + dg \\ u &= cf + dh \end{aligned}\right\} \text{ 4 phép cộng ma trận con cấp } (n/2) \times (n/2)$$

Thuật toán chia để trị

Ý tưởng: ma trận $n \times n = 2 \times 2$ ma con trận cấp $(n/2) \times (n/2)$:

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

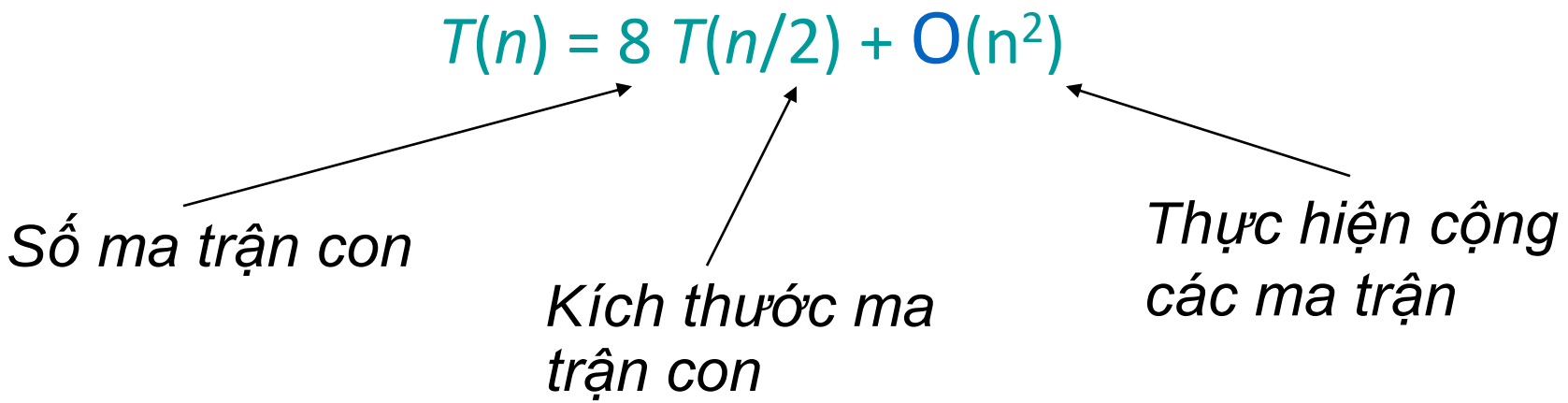
$$C = A.B$$

$$\left.\begin{aligned} r &= ae + bg \\ s &= af + bh \end{aligned}\right\}$$ Đệ quy

8 phép nhân ma trận con cấp $(n/2) \times (n/2)$

$$\left.\begin{aligned} t &= ce + dg \\ u &= cf + dh \end{aligned}\right\}$$ 4 phép cộng ma trận con cấp $(n/2) \times (n/2)$

Phân tích thuật toán chia để trị



Phân tích thuật toán chia để trị

$$T(n) = 8\,T(n/2) + O(n^2)$$

Số ma trận con

Kích thước ma trận con

Thực hiện cộng các ma trận

$$\alpha = 2 < \log_2 8 = 3 \Rightarrow \text{TH1} \Rightarrow T(n) = O(n^{\log_2 8}) = O(n^3)$$

Phân tích thuật toán chia để trị

$$T(n) = 8\,T(n/2) + O(n^2)$$

Số ma trận con

Kích thước ma trận con

Thực hiện cộng các ma trận

$$\alpha = 2 < \log_2 8 = 3 \Rightarrow \text{TH1} \Rightarrow T(n) = O(n^{\log_2 8}) = O(n^3)$$

Không tốt hơn thuật toán ban đầu!

Ý tưởng của Strassen

- Nhân 2×2 ma trận chỉ với 7 phép nhân đệ quy

Ý tưởng của Strassen

- Nhân 2×2 ma trận chỉ với 7 phép nhân đệ quy

$P_1 = a \,.\, (f - h)$

$P_2 = (a + b) \,.\, h$

$P_3 = (c + d).\, e$

$P_4 = d \,.\, (g - e)$

$P_5 = (a + d) \,.\, (e + h)$

$P_6 = (b - d) \,.\, (g + h)$

$P_7 = (a - c) \,.\, (e + f)$

Ý tưởng của Strassen

- Nhân 2×2 ma trận chỉ với 7 phép nhân đệ quy

$P_1 = a \,.\, (f - h)$

$P_2 = (a + b) \,.\, h$

$P_3 = (c + d).\, e$

$P_4 = d \,.\, (g - e)$

$P_5 = (a + d) \,.\, (e + h)$

$P_6 = (b - d) \,.\, (g + h)$

$P_7 = (a - c) \,.\, (e + f)$

$$r = P_5 + P_4 - P_2 + P_6$$

$$s = P_1 + P_2$$

$$t = P_3 + P_4$$

$$u = P_5 + P_1 - P_3 - P_7$$

Ý tưởng của Strassen

- Nhân 2×2 ma trận chỉ với 7 phép nhân đệ quy

$P_1 = a \,.\, (f - h)$

$P_2 = (a + b) \,.\, h$

$P_3 = (c + d).\, e$

$P_4 = d \,.\, (g - e)$

$P_5 = (a + d) \,.\, (e + h)$

$P_6 = (b - d) \,.\, (g + h)$

$P_7 = (a - c) \,.\, (e + f)$

$$r = P_5 + P_4 - P_2 + P_6$$

$$s = P_1 + P_2$$

$$t = P_3 + P_4$$

$$u = P_5 + P_1 - P_3 - P_7$$

7 phép nhân, 18 phép cộng/trừ

Chú ý: không phụ thuộc vào sự giao hoán của phép nhân!

Ý tưởng của Strassen

- Nhân 2×2 ma trận chỉ với 7 phép nhân đệ quy

$P_1 = a \,.\, (f - h)$

$P_2 = (a + b) \,.\, h$

$P_3 = (c + d).\, e$

$P_4 = d \,.\, (g - e)$

$P_5 = (a + d) \,.\, (e + h)$

$P_6 = (b - d) \,.\, (g + h)$

$P_7 = (a - c) \,.\, (e + f)$

$$
\begin{aligned}
r &= P5 + P4 - P2 + P6 \\
&= (a + d)(e + h) \\
&+ d(g - e) - (a + b)h \\
&+ (b - d)(g + h) \\
&= ae + ah + de + dh \\
&+ dg - de - ah - bh \\
&+ bg + bh - dg - dh \\
&= ae + bg
\end{aligned}
$$

Thuật toán Strassen

1. Divide: Chia A và B thành các ma trận con cấp $(n/2)\times(n/2)$. Xây dựng các MT số hạng sử dụng $+$ và $-$.

2. Conquer: Thực hiện 7 phép nhân (đệ quy) ma trận con cấp $(n/2)\times(n/2)$.

3. Combine: Xây dựng C sử dụng $+$ và $-$ dựa trên các ma trận con cấp $(n/2)\times(n/2)$.

Thuật toán Strassen

1. Divide: Chia A và B thành các ma trận con cấp $(n/2)\times(n/2)$. Xây dựng các MT số hạng sử dụng $+$ và $-$.

2. Conquer: Thực hiện 7 phép nhân (đệ quy) ma trận con cấp $(n/2)\times(n/2)$.

3. Combine: Xây dựng C sử dụng $+$ và $-$ dựa trên các ma trận con cấp $(n/2)\times(n/2)$.

$$T(n) = 7T(n/2) + O(n^2)$$

Phân tích thuật toán Strassen

$$T(n) = 7\,T(n/2) + O(n^2)$$

Phân tích thuật toán Strassen

$$T(n) = 7T(n/2) + O(n^2)$$

$$\alpha = 2 < \log_b a = \log_2 7 \approx 2.8074 \Rightarrow \text{TH1} \Rightarrow T(n) = O(n^{lg7}) \approx O(n^{2.8074})$$

Phân tích thuật toán Strassen

$$T(n) = 7\,T(n/2) + O(n^2)$$

$\alpha = 2 < \log_b{}^a = \log_2{}^7 \approx 2.8074$ => TH1 => $T(n) = O(n^{lg7}) \approx O(n^{2.8074})$

Con số 2.8074 có thể không nhỏ hơn nhiều so với 3, nhưng sự khác biệt là ở số mũ (ảnh hưởng rất đáng kể tới thời gian chạy). Thực tế, thuật toán Strassen giao với thuật toán ban đầu với $n \geq 32$.

Phân tích thuật toán Strassen

$$T(n) = 7\,T(n/2) + O(n^2)$$

$\alpha = 2 < \log_b{}^a = \log_2{}^7 \approx 2.8074$ => TH1 => $T(n) = O(n^{lg7}) \approx O(n^{2.8074})$

Con số 2.8074 có thể không nhỏ hơn nhiều so với 3, nhưng sự khác biệt là ở số mũ (ảnh hưởng rất đáng kể tới thời gian chạy). Thực tế, thuật toán Strassen giao với thuật toán ban đầu với $n \geq 32$.

Tốt nhất cho đến nay : $O(n^{2.376\ldots})$.

Phương pháp quay lui

Bài 5: **Biểu thức zero**. Cho một số tự nhiên N ≤ 9. Giữa các số từ 1 đến N hãy thêm vào các dấu + và - sao cho kết quả thu được bằng 0. Hãy viết chương trình tìm tất cả các khả năng có thể.

→ Áp dụng thuật toán đệ quy quay lui để giải quyết bài toán này!!!

Phương pháp quay lui

Bài 5: **Biểu thức zero**.

Giả sử ta đã điền các dấu’+’ và ’-’ vào các số từ 1 đến i, bây giờ cần điền các dấu vào giữa i và i + 1

→ chọn một trong ba khả năng:

1. điền dấu ’+’
2. điền dấu ’-’
3. không điền dấu nào cả

Phương pháp quay lui

Bài 5: **Biểu thức zero**.

- Sau khi chọn một trong ba khả năng trên, tiếp tục lựa chọn dấu để điền vào giữa i + 1 và i + 2 bằng cách gọi đệ quy
- Nếu i = N - 1 ta sẽ kiểm tra xem cách điền đó có thoả mãn kết quả bằng 0 hay không
  - Nếu tổng đúng bằng 0 thì cách điền đó là một nghiệm của bài toán, ta ghi nhận nó
  - Nếu i < N - 1 thì tiếp tục gọi đệ quy

Phương pháp quay lui

Bài 6: Cho một tự nhiên N ≤ 30. Tìm tất cả các cách phân tích số N thành tổng của các số nguyên dương. Các cách phân tích là hoán vị của nhau thì chỉ tính là một cách.

Phương pháp quay lui

Bài 6:

- Sử dụng mảng *x* để lưu nghiệm của bài toán
- Mảng *t* được xây dựng như sau: mỗi t[i] là tổng các phần tử trong mảng *x* từ x[1] đến x[i]: t[i] = x[1] + x[2] + … + x[i]
- Thêm ràng buộc x[i-1] ≤ x[i] để tránh bị trùng lặp khi liệt kê các dãy *x* có tổng các phần tử đúng bằng N
- Thủ tục đệ quy sẽ thử tất cả các giá trị có thể nhận của x[i] với x[i-1] ≤ x[i]

Phương pháp quay lui

Bài 6:

- Mỗi nghiệm *x* có số phần tử khác nhau nên phải thêm tham số thể hiện số phần tử của *x*. Vậy khi nào thì in kết quả và khi nào thì đệ quy tiếp?
- Nhớ rằng t[i-1] lưu tổng tất cả các phần tử từ x[1] đến x[i-1], do đó:
  - Khi t[i] = N, tức là x[i] = N – t[i-1] thì in kết quả
  - Khi tìm tiếp, x[i+1] phải lớn hơn hoặc bằng x[i]. Mặt khác, t[i+1] là tổng của các số từ x[1] đến x[i+1] không vượt quá N. Ta có: t[i+1] ≤ N ⇔ t[i-1] + x[i] + x[i+1] ≤ N ⇔ x[i] + x[i+1] ≤ N – t[i-1], tức là x[i] ≤ (N – t[i-1])/2

    Ví dụ: với N = 10 thì x[1] = 6, 7, 8, 9 là vô nghĩa vì không chọn được x[2]

Phương pháp quay lui

Bài 6:

- Ta gọi đệ quy tìm tiếp khi giá trị x[i] được chọn còn cho phép chọn thêm một phần tử khác lớn hơn hoặc bằng nó mà không làm tổng vượt quá N.
- In kết quả khi x[i] đúng bằng số thiếu hụt của tổng i-1 phần tử đầu tiên so với N (tức bằng x[i] = N – t[i-1])

Phương pháp quay lui

- **Bài 7**: Bài toán cái ba lô
  - Cho một cái ba lô có thể đựng một trọng lượng *W* và *n* loại đồ vật, mỗi đồ vật *i* có một trọng lượng *g*[i] và một giá trị *v*[i]. Tất cả các loại đồ vật đều có số lượng không hạn chế. Tìm một cách lựa chọn các đồ vật đựng vào ba lô, chọn các loại đồ vật nào, mỗi loại lấy bao nhiêu sao cho tổng trọng lượng không vượt quá *W* và tổng giá trị là lớn nhất

Phương pháp quay lui

- Bài toán cái ba lô
  - Gọi $X = (X_1, X_2, …., X_n)$ với X_i là số nguyên không âm là một phương án. X_i là số đồ vật thứ i
  - Cần tìm X sao cho:
    - $X_1 \times g[1] + X_2 \times g[2] + … + X_n \times g[n] \leq W$
    - $F(X) = X_1 \times v[1] + X_2 \times v[2] + … + X_n \times v[n]$ --> Max

Phương pháp quay lui

- Bài toán cái ba lô

```
void try (x[], int i) {
        for (int j = 0; j <= n; j++){
                x[i] = j;
                if (i == n) {
                        if (x[1] * g[1] + x[2] * g[2] + … + x[n] * g[n] ≤ w)
                                printResult();
                } else try (x, i+1);
        }
}
```

Phương pháp quay lui

Bài 8: Dãy con tăng dài nhất

- Cho một mảng n phần tử $A[1, 2, \ldots, n]$. Tìm một dãy dài nhất các chỉ số $1 \le i_1 < i_2 < \ldots < i_k \le n$ sao cho $A[i_1] \le A[i_2] \le \ldots \le A[i_k]$.
- Ví dụ: $A[10] = \{1, 6, 5, 4, 7, 8, 2, 9, 0, 10\}$. Một trong số dãy con tăng dài nhất là 1, 6, 7, 8, 9, 10 với chiều dài 6.

Phương pháp quay lui

Bài 8: Dãy con tăng dài nhất

- Dãy và dãy con
  - Một dãy số nguyên là rỗng hoặc là một số nguyên theo sau bởi một dãy các số nguyên

Phương pháp quay lui

Bài 8: Dãy con tăng dài nhất

- Chiến lược đệ quy
  - Dãy con của một dãy rỗng là một dãy rỗng
  - Dãy con của $A[1, 2, \ldots, n]$ là hoặc là một dãy con $A[2, \ldots, n]$ hoặc là A[1] theo sau bởi $A[2, \ldots, n]$

Phương pháp quay lui

Bài 8: Dãy con tăng dài nhất

- Chiến lược đệ quy
  - Do chỉ tìm dãy con tăng dài nhất nên ta loại bỏ trường hợp dãy rỗng

    Dãy con tăng dài nhất LIS của A[1 .. n]
    hoặc là LIS của A[2 .. n]
    hoặc là A[1] theo sau bởi LIS của A[2 .. n] với các phần tử lớn hơn A[1]

Phương pháp quay lui

Bài 8: Dãy con tăng dài nhất

- Chiến lược đệ quy
  - Dãy con tăng dài nhất với các phần tử lớn hơn x

If A[1] ≤ x, dãy con tăng dài nhất LIS của A[1 .. n] với các phần tử lớn hơn x
là dãy con tăng dài nhất LIS của A[2 .. n] với các phần tử lớn hơn x

Ngược lại, dãy con tăng dài nhất LIS của A[1 .. n] với các phần tử lớn hơn x
hoặc là dãy con tăng dài nhất LIS của A[2 .. n] với các phần tử lớn hơn x
hoặc là A[1] theo sau bởi LIS của A[2 .. n] với các phần tử lớn hơn A[1]

Phương pháp quay lui

Bài 8: Dãy con tăng dài nhất

- Chiến lược đệ quy
  - Dãy con tăng dài nhất với các phần tử lớn hơn x = $-\infty$

```
LIS (A[1 .. n]) {
        return LISBIGGER (−∞, A[1 .. n])
}
```

Phương pháp quay lui

Bài 8: Dãy con tăng dài nhất

- Chiến lược đệ quy
  - Dãy con tăng dài nhất với các phần tử lớn hơn x = $-\infty$

```
LISBIGGER(prev, A[1 .. n]) {
    if n = 0
        return 0
    else {
        max ← LISBIGGER(prev, A[2 .. n])
        if A[1] > prev
            L ← 1 + LISBIGGER(A[1], A[2 .. n])
            if L > max
                max ← L
        return max
    }
}
```

Phương pháp tham lam

Bài 9: Máy rút tiền ATM

- Giả sử có các loại tiền: 100.000 vnđ, 50.000 vnđ, 40.000 vnđ và 10.000 vnđ
- Mỗi loại tiền đều có số lượng không hạn chế
- Khách hàng cần rút một số tiền ***n*** vnđ (tính chẵn đến 10.000 vnđ, hay ***n*** chia hết cho 10.000)
- Phương án trả tiền sao cho trả đủ ***n*** vnđ và số tờ tiền phải trả là ít nhất

Phương pháp tham lam

Bài 9: Máy rút tiền ATM

- Gọi X_1, X_2, X_3, X_4 lần lượt là mệnh giá các đồng tiền 100.000 vnđ, 50.000 vnđ, 40.000 vnđ và 10.000 vnđ
- Ta phải tìm bộ X_1, X_2, X_3, X_4 sao cho:

  $100.000 * X_1 + 50.000 * X_2 + 40.000 * X_3 + 10.000 * X_4 = n$

  và $X_1 + X_2 + X_3 + X_4 \rightarrow \min$

Phương pháp tham lam

Bài 9: Máy rút tiền ATM

- Trả ra các tờ tiền có mệnh giá 100.000 nhiều nhất có thể sao cho tổng giá trị không vượt quá n (tổng số tiền cần phải trả ra).
  Số tờ tiền mệnh giá 100.000 cần trả ra là: $X_1 = n / 100.000$.
  Số tiền còn lại cần phải trả ra là $n = n - 100.000 * X_1$
- Tiếp theo, trả ra các tờ tiền có mệnh giá 50.000 nhiều nhất có thể sao cho tổng giá trị không vượt quá số tiền còn lại phải trả ra là n
  Số tờ tiền mệnh giá 50.000 cân trả ra là: $X_2 = n / 50.000$.
  Số tiền còn lại cần phải trả ra là $n = n - 50.000 * X_2$.
- Tương tự ta có:
  Số tờ tiền mệnh giá 40.000 cần trả ra là: $X_3 = n / 40.000$; $n = n - 40.000 * X_3$.
  Số tờ tiền mệnh giá 10.000 cần trả ra là: $X_4 = n / 10.000$.

Phương pháp tham lam

Bài 10: Lập lịch họp

- Có *n* cuộc họp, cuộc họp thứ *i* bắt đầu vào thời điểm S_i và kết thúc ở thời điểm F_i. Do chỉ có một phòng hội thảo nên hai cuộc họp bất kì sẽ được cùng bố trí phục vụ nếu khoảng thời gian làm việc của chúng chỉ giao nhau tại đầu mút. Hãy bố trí phòng họp để phục vụ được nhiều cuộc họp nhất.

Cuộc họp	Bắt đầu (giờ)	Kết thúc (giờ)
1	6	7
2	7	9
3	8	10
4	10	12
5	9	11

Phương pháp tham lam

Bài 11: Cho thuê máy

- Trung tâm tính toán hiệu năng cao nhận được đơn đặt hàng của n khách hàng. Khách hàng i muốn sử dụng máy trong khoảng thời gian từ a_i đến b_i và trả tiền thuê là c_i. Hãy bố trí lịch cho thuê máy để tổng số tiền thu được là lớn nhất mà thời gian sử dụng máy của hai khách hàng bất kì được phục vụ đều không giao nhau (cả trung tâm chỉ có một máy cho thuê).

Khách hàng	Bắt đầu (ngày)	Kết thúc (ngày)	Tiền thuê
1	6	7	15
2	8	10	27
3	9	12	43
4	11	18	110
5	15	21	90

Phương pháp tham lam

Bài 12: Bài toán đóng thùng

- Input:
  - Các thùng cùng kích thước là B
  - N số nguyên dương là số đồ vật
  - Các đồ vật $A_1, \ldots, A_N$, kích thước vật $A_i \leq B$.
- Output
  - Số thùng ít nhất chứa tất cả các vật
- Minh họa các bước của thuật toán với B = 6, N = 6 với các đồ vật có kích thước là 3, 2, 5, 1, 4, 3

Phương pháp tham lam

Bài 12: Bài toán đóng thùng

- Minh họa các bước của thuật toán với B = 6, N = 6 với các đồ vật có kích thước là 3, 2, 5, 1, 4, 3
- **Cách 1**: tham lam theo kích thước đồ vật bé nhất trước.
  - Thùng 1: (1, 2, 3),
  - Thùng 2: (3),
  - Thùng 3: (4),
  - Thùng 4: (5).
- **Cách 2**: tham lam theo kích thước đồ vật lớn nhất trước.
  - Thùng 1: (5, 1),
  - Thùng 2: (4, 2),
  - Thùng 3: (3, 3).

Phương pháp tham lam

Bài 13: Bài toán người du lịch

- Có n thành phố được đánh số từ 1 đến n. Một người du lịch, xuất phát từ thành thành phố s, muốn đi thăm tất cả các thành phố khác, mỗi thành phố đúng một lần, rồi lại quay về nơi xuất phát. Giả thiết biết chi phí đi từ thành phố i đến thành phố j là c(i, j), $1 \leq i, j \leq n$. Hãy tìm một hành trình cho người du lịch sao cho chi phí của hành trình này là nhỏ nhất.

Phương pháp tham lam

Bài 13: Bài toán người du lịch

- Bắt đầu từ thành phố xuất phát s, gán x_1 bằng s,
- Tìm thành phố tiếp theo có chi phí đi từ s đến đó là ít nhất (tìm giá trị nhỏ nhất của các c(s, j), j ≠ s) và gán x_2 bằng số hiệu của thành phố này, ...
- Một cách tổng quát, từ thành phố x_{i-1}, tìm thành phố x_i là thành phố chưa đến, có chi phí đi từ x_{i-1} là ít nhất (i = 1, 2, ..., n).
- Hành trình $x_1, x_2, ..., x_n$ là kết quả cần tìm.
- Rõ ràng không thể khẳng định đây là hành trình tối ưu mặc dù nó được lựa chọn tốt nhất trong từng bước.

Phương pháp quy hoạch động

Bài 14: Đổi tiền

- Ở đất nước Omega người ta chỉ tiêu tiền xu. Có N loại tiền xu, loại thứ i có mệnh giá là A_i đồng. Một người khách du lịch đến Omega du lịch với số tiền M đồng. Ông ta muốn đổi số tiền đó ra tiền xu Omega để tiện tiêu dùng. Ông ta cũng muốn số đồng tiền đổi được là ít nhất (cho túi tiền đỡ nặng khi đi đây đi đó). Bạn hãy giúp ông ta tìm cách đổi tiền.

Phương pháp quy hoạch động

Bài 14: Đổi tiền

- Hướng dẫn
  - Bài toán này giống bài toán xếp ba lô. Ở đây, “khối lượng” là mệnh giá, “giá trị” là 1 và tổng giá trị yêu cầu là nhỏ nhất.
  - Do đó ta cũng xây dựng bảng phương án và công thức truy hồi một cách tương tự

Phương pháp quy hoạch động

Bài 14: Đổi tiền

- Hướng dẫn
  - Gọi L[i, t] là số đồng xu ít nhất nếu đổi t đồng ra i loại tiền xu (từ 1 đến i).
  - Với giới hạn số tiền cần đổi là t, việc chọn trong số các tiền xu từ 1 đến i để có số đồng xu là ít nhất sẽ là một trong hai khả năng:
    - Nếu không chọn tiền xu thứ i thì L[i, t] là giá trị nhỏ nhất có thể có bằng cách chọn trong số các tiền xu {1, 2, …, i-1} với giới hạn số tiền t. Tức là L[i, t] = L[i - 1, t].
    - Nếu chọn tiền xu thứ i thì L[i, t] bằng với mệnh giá tiền xu thứ i (= A[i]) cộng với giá trị nhỏ nhất có thể có được bằng cách chọn trong số các tiền xu từ 1 đến i (có thể được chọn nhiều lần) với giới hạn số tiền là t−A[i] (= L[i, t−A[i]]).
    - Vì theo cách xây dựng L[i, t] là **nhỏ nhất có thể** nên L[i, t] là **min** của hai trường hợp trên.

Phương pháp quy hoạch động

Bài 14: Đổi tiền

- Hướng dẫn
  - Công thức tính L[i, t] như sau:
  - L[i, 0] = 0.
  - L[0, t] = +∞.
  - L[i, t] = L[i−1, t] nếu t < A[i].
  - L[i, t] = min(L[i−1, t], L[i, t−A[i]] + 1) nếu t ≥ A[i].

Phương pháp quy hoạch động

Bài 14: Đổi tiền

- Nhận xét: Ta thấy rằng, dòng i chỉ phụ thuộc vào dòng i - 1. Do đó, chỉ cần lưu dòng i - 1 (vào mảng P) và dòng i (vào mảng L).
  - P[t] = +∞ với i t >= 0.
  - L[t] = P[t] nếu t < A[i].
  - L[t] = min(P[t], L[t−A[i]] + 1) nếu t ≥ A[i].

Phương pháp quy hoạch động

Bài 15: Xâu con chung dài nhất

- Cho 2 xâu X, Y. Hãy tìm xâu con của X và của Y có độ dài lớn nhất. Biết xâu con của một xâu thu được khi xóa một số kí tự thuộc xâu đó (hoặc không xóa kí tự nào).
- Ví dụ: X = “ALTRUISTIC” và Y = “ALGORITHMS” có xâu con chung dài nhất là “ALRIT” với độ dài là 5.

Phương pháp quy hoạch động

Bài 15: Bài toán dễ hơn tìm độ dài của xâu con chung dài nhất

- Gọi L[i, j] là độ dài xâu con chung dài nhất của xâu X_i gồm i kí tự phần đầu của X (X_i = X[1..i]) và xâu Y_j gồm j kí tự phần đầu của Y (Y_j = Y[1..j]).
- Dễ dàng thấy được rằng, lời giải của mỗi bài toán con sẽ phụ thuộc vào i và j, L[i, j].
- Bài toán lớn sẽ được giải bằng cách lần lượt giải các bài toán con xuất phát từ L[0, 0] và tăng dần độ dài xâu được lấy ra cho đến khi chúng ta lấy ra toàn bộ xâu ban đầu.

Phương pháp quy hoạch động

Bài 15: Bài toán dễ hơn tìm độ dài của xâu con chung dài nhất

- Giải các bài toán con:
  - Nếu một trong hai xâu là rỗng thì xâu con chung của chúng cũng là rỗng

    → L[0, j] = L[i, 0] = 0 (cơ sở của quy hoạch động)
  - Nếu i > 0 và j > 0 thì
    - Nếu ký tự cuối cùng của xâu X_i không có mặt trong xâu con chung dài nhất thì nó có thể bị bỏ qua mà không ảnh hưởng gì đến kết quả → L[i, j] = L[i - 1, j].
    - Tương tự, ký tự cuối cùng của xâu Y_j không ảnh hưởng đến kết quả thì → L[i, j] = L[i, j-1].
    - Hai ký tự cuối cùng của hai xâu X_i và Y_j đều có mặt trong xâu con chung dài nhất (X[i] = Y[j]) → L[i, j] = L[i−1, j−1] + 1 nếu $X_i = Y_j$

Phương pháp quy hoạch động

Bài 15: Bài toán dễ hơn tìm độ dài của xâu con chung dài nhất

- Ta có công thức truy hồi cho quy hoạch động như sau:
  - L[0, j] = L[i, 0] = 0 (trường hợp cơ sở)
  - L[i, j] = L[i−1, j−1] + 1 nếu X[i] = Y[j]
  - L[i, j] = max(L[i−1, j], L[i, j−1]) nếu X[i] ≠ Y[j].

Phương pháp quy hoạch động

Bài 15: Bài toán dễ hơn tìm độ dài của xâu con chung dài nhất

- Nhận xét: để tính L[i, j] của bảng phương án, chỉ cần 3 ô L[i−1, j−1], L[i−1, j] và L[i, j−1]. Tức, tính dòng i chỉ cần biết dòng i – 1.
- Chỉ cần hai mảng một chiều lưu dòng i – 1 (vào mảng P) và dòng i (vào mảng L).
  - P[j] = 0
  - L[j] = P[j−1] + 1 nếu X[i] = Y[j]
  - L[j] = max(P[j], L[j−1]) nếu X[i] ≠ Y[j].

Phương pháp quy hoạch động

Bài 16: Hai nước Alpha và Beta nằm ở hai bên bờ sông Omega, Alpha nằm ở bờ bắc và có M thành phố được đánh số từ 1 đến M, Beta nằm ở bờ nam và có N thành phố được đánh số từ 1 đến N (theo vị trí từ đông sang tây). Mỗi thành phố của nước này thường có quan hệ kết nghĩa với một số thành phố của nước kia. Để tăng cường tình hữu nghị, hai nước muốn xây các cây cầu bắc qua sông, mỗi cây cầu sẽ là nhịp cầu nối 2 thành phố kết nghĩa. Với yêu cầu là các cây cầu không được cắt nhau và mỗi thành phố chỉ là đầu cầu cho nhiều nhất là một cây cầu, hãy chỉ ra cách bắc cầu được nhiều cầu nhất.

Phương pháp quy hoạch động

Bài 16: Gợi ý

Gọi các thành phố của Alpha lần lượt là $A_1, A_2, \ldots, A_M$; các thành phố của Beta là $B_1, B_2, \ldots, B_N$.

- Nếu thành phố A_i và B_j kết nghĩa với nhau thì coi A_i "bằng" B_j.
- Để các cây cầu không cắt nhau, nếu ta đã chọn cặp thành phố (A_i, B_j) để xây cầu thì cặp tiếp theo phải là cặp (A_u, B_v) sao cho $u > i$ và $v > j$.

→ Các cặp thành phố được chọn xây cầu có thể coi là một dãy con chung của hai dãy A và B.

→ Bài toán tìm dãy con chung dài nhất, ở đây hai phần tử "bằng" nhau nếu chúng có quan hệ kết nghĩa.