Chương 5: Kỹ thuật Chia để trị (Divide-and-Conquer)

Giới thiệu chung

Chia để trị là kỹ thuật thiết kế giải thuật rất thông dụng, gồm 3 bước tiến hành:

Bước 1. Thể hiện của vấn đề được chia thành nhiều thể hiện nhỏ hơn (nhưng vẫn duy trì bản chất của vấn đề).

Bước 2. Tiếp tục giải quyết những thể hiện nhỏ hơn theo cách của Bước 1 (đệ qui) cho đến khi thể hiện đủ nhỏ thì xử lý thật sự.

Bước 3. Lời giải của những thể hiện nhỏ hơn sẽ được tổ hợp lại dần để cuối cùng cho ra lời giải của thể hiện ban đầu của vấn đề.

Ví dụ: Tìm số lớn nhất của dãy n số.

Chú ý: Giảm để trị là một nhánh tách ra từ Chia để trị, hiện đã được thừa nhận rộng rãi như là một chiến lược thiết kế giải thuật độc lập.

Hệ thức truy hồi chia để trị

Gọi n là kích thước nguyên thủy của thể hiện của bài toán. Chia bài toán thành a (≥ 2) bài toán con, mỗi cái có kích thước n/b (trường hợp lý tưởng là n/a) với $b > 1$. Hệ thức truy hồi chia để trị, biểu diễn thời gian thực thi $\mathcal{T}(n)$:

$$\mathcal{T}(n) = a\mathcal{T}(n/b) + f(n)$$

với $f(n)$ là hàm tính thời gian dành cho việc phân chia và tổng hợp kết quả.

Định lý chủ (*Master theorem*)

Cho hệ thức truy hồi chia để trị $\mathcal{T}(n) = a\mathcal{T}(n/b) + f(n)$. Nếu $f(n) \in \Theta(n^d)$ với $d \geq 0$:

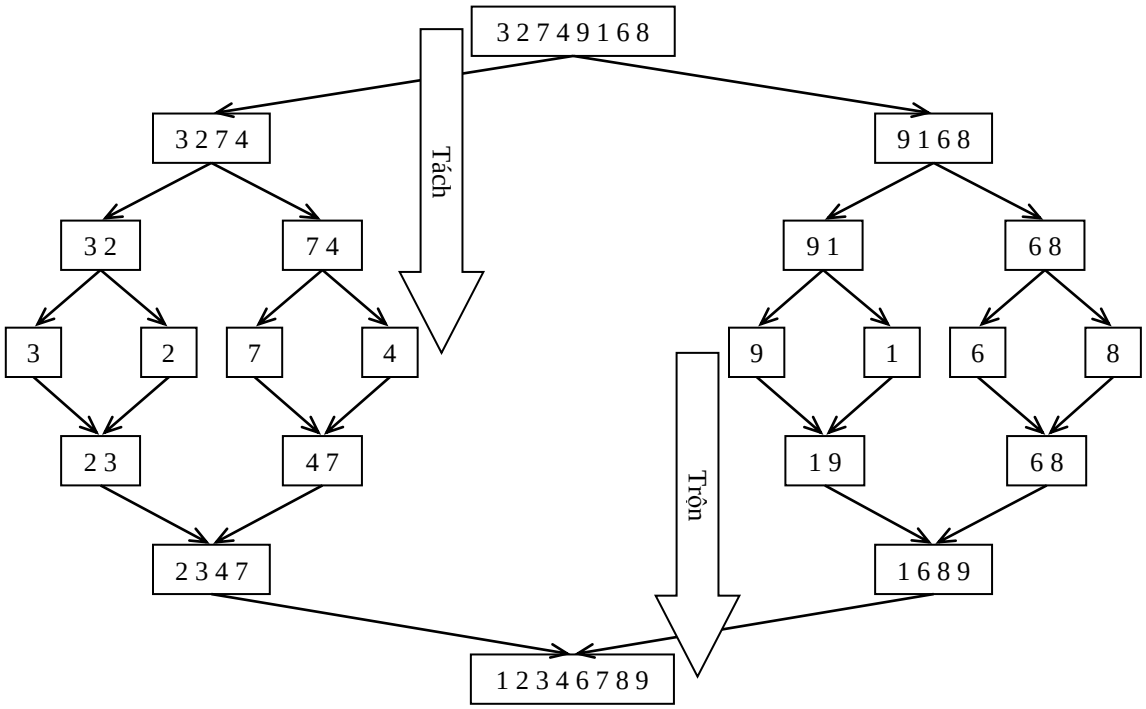
$$T(n) \in \begin{cases} \Theta(n^d) & \text{nếu } a < b^d \\ \Theta(n^d \log n) & \text{nếu } a = b^d \\ \Theta\left(n^{\log_b a}\right) & \text{nếu } a > b^d \end{cases}$$

Kết quả thu được tương tự đối với O và Ω.

Ví dụ: Tìm số lớn nhất của dãy n số.

Ví dụ: Tìm đồng thời số nhỏ nhất và lớn nhất của mảng một chiều có n phần tử.

Sắp xếp trộn (Mergesort)



Giải thuật

```
mergeSort(a[1..n], low, high) {
   if (low < high) {
      mid = ⌊(low + high) / 2⌋;
      mergeSort(a, low, mid);
      mergeSort(a, mid + 1, high);
      merge(a, low, mid, high);
   }
}
merge(a[1..n], low, mid, high) {
   i = low;     j = mid + 1;
   k = low;
   while (i ≤ mid) && (j ≤ high)
      if (a[i] ≤ a[j])
         buf[k ++] = a[i ++];
      else
         buf[k ++] = a[j ++];
   if (i > mid)
      buf[k .. high] = a[j .. high];
   else
      buf[k .. high] = a[i .. mid];
   a[low .. high] = buf[low .. high];
}
mergeSort(a, 1, n);
```

Sắp xếp Nhanh (Quicksort)

Giải thuật

```
Quicksort(a[left .. right]) {
   if (left < right){
      s = Partition(a[left .. right]);

      Quicksort(a[left .. s - 1]);
      Quicksort(a[s + 1 .. right]);
   }
}

Partition(a[left .. right]) {
   p = a[left];

   i = left + 1;
   j = right;
   do {
      while (a[i] < p)
         i++;
      while (a[j] > p)
         j--;
      swap(a[i], a[j]);
   } while (i < j);

   swap(a[i], a[j]);
   swap(a[left], a[j]);
   return   j;
}
```

Chú ý: Đoạn mã giả nêu trên có thể khiến cho *i* vượt giới hạn bên phải.

Đánh giá:

Trường hợp tốt nhất: Khi *pivot* luôn là giá trị trung vị

$$C_b(n) \in \Theta(n \log n)$$

Trường hợp xấu nhất: Khi *pivot* luôn là giá trị nhỏ hoặc lớn nhất

$$C_w(n) \in \Theta(n^2)$$

Trường hợp trung bình: $C_a(n) \approx 1.39n \log_2 n$

Nhân số lớn

Ý tưởng: Gauss nhận thấy, khi nhân hai số phức

$$(a + bi)(c + di) = (ac - bd) + (bc + ad)i$$

và biết rằng:

$$bc + ad = (a + b)(c + d) - (ac + bd)$$

Ví dụ: Xét phép nhân của hai số 12 × 34. Hai số này có thể biểu diễn là:

$12 = 1 \times 10^1 + 2 \times 10^0$ và $34 = 3 \times 10^1 + 4 \times 10^0$

$$12 \times 34 = (1 \times 10^1 + 2 \times 10^0) \times (3 \times 10^1 + 4 \times 10^0)$$
$$= (1 \times 3) \times 10^2 + (1 \times 4 + 2 \times 3) \times 10^1 + (2 \times 4) \times 10^0$$

Dựa vào ý tưởng của Gauss, có thể thay thế:

$$1 \times 4 + 2 \times 3 = (1 + 2) \times (3 + 4) - (1 \times 3 + 2 \times 4) = 10$$

Phát biểu: Giả sử đã xây dựng kiểu dữ liệu `large_integer` với các phép tính: `mul` 10^m, `div` 10^m, `mod` 10^m với chi phí tuyến tính. Hãy thực hiện phép nhân hai số có n chữ số (với n bất kỳ và lớn hơn ngưỡng α): $u \times v$.

Giải thuật

```
large_integer MUL(large_integer u, v) {
   large_integer x, y, w, z;
   n = max(Số chữ số của u, Số chữ số của v);
   if (u == 0 || v == 0)
      return 0;
   else
      if (n == α)
         return u * v;  // built-in operator
      else {
         m = ⌊n / 2⌋;
         x = u div 10^m;  y = u mod 10^m;
         w = v div 10^m;  z = v mod 10^m;
         return   MUL(x, w) mul 10^2m +
                 (MUL(x, z) + MUL(y, w)) mul 10^m +
                  MUL(y, z);
      }
}
```

Đánh giá: Hệ thức truy hồi chia để trị là:

$$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n) \text{ với } T(1) = 1$$

Dựa vào định lý chủ: $T(n) \in \Theta(n^2)$

Giải thuật (cải tiến)

```
large_integer MUL(large_integer u, v, n) {
   n = max(Số chữ số của u, Số chữ số của v);
   if (u == 0 || v == 0)      return 0;
   else
      if (n == α)             return u * v;
      else {
         m = ⌊n / 2⌋;
         x = u div 10^m;  y = u mod 10^m;
         w = v div 10^m;  z = v mod 10^m;
         r = MUL(x + y, w + z);
         p = MUL(x, w);
         q = MUL(y, z);
         return p mul 10^2m + (r − p − q) mul 10^m + q;
      }
}
```

Đánh giá: Hệ thức truy hồi chia để trị là:

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n) \in \Theta(n^{1.585})$$

Mở rộng: Nhân hai số nguyên dương *n* bit (cho rằng, *n* là lũy thừa của 2).

Gọi *x* và *y* là hai số nguyên *n* bit. Gọi x_L, x_R lần lượt là số nguyên tương ứng với *n*/2 bit bên trái và *n*/2 bit bên phải của *x*. Tương tự, ta cũng có y_L và y_R.

$$x = 2^{n/2}x_L + x_R$$
$$y = 2^{n/2}y_L + y_R$$

Tích $x \times y$ có thể được viết như sau:

$$x \times y = (2^{n/2}x_L + x_R)(2^{n/2}y_L + y_R) = 2^n\, x_L\, y_L + 2^{n/2}\,(x_L\, y_R + x_R\, y_L) + x_R\, y_R$$

Giải thuật

```
int   multiply(x, y) {
   n = max(|x|_bit , |y|_bit );
   if (n == α)    return      x × y;

   x_L = ⌈n / 2⌉ bit bên trái của x;
   x_R = ⌊n / 2⌋ bit bên phải của x;
   y_L = ⌈n / 2⌉ bit bên trái của y;
   y_R = ⌊n / 2⌋ bit bên phải của y;

   P1 = multiply(x_L, y_L);
   P2 = multiply(x_R, y_R);
   P3 = multiply(x_L + x_R, y_L + y_R);
   return   P1 × 2^n + (P3 − P1 − P2) × 2^(n/2) + P2;
}
```

Nhân ma trận (của) Strassen

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$
$$= \begin{bmatrix} a_{11} \times b_{11} + a_{12} \times b_{21} & a_{11} \times b_{12} + a_{12} \times b_{22} \\ a_{21} \times b_{11} + a_{22} \times b_{21} & a_{21} \times b_{12} + a_{22} \times b_{22} \end{bmatrix}$$

Như vậy, số lượng phép × phải thực hiện là 8 và phép + là 4.

Tuy nhiên, ma trận kết quả có thể tìm được như sau:

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

với:

$m_1 = (a_{11} + a_{22}) \times (b_{11} + b_{22})$
$m_2 = (a_{21} + a_{22}) \times b_{11}$
$m_3 = a_{11} \times (b_{12} - b_{22})$
$m_4 = a_{22} \times (b_{21} - b_{11})$
$m_5 = (a_{11} + a_{12}) \times b_{22}$
$m_6 = (a_{21} - a_{11}) \times (b_{11} + b_{12})$
$m_7 = (a_{12} - a_{22}) \times (b_{21} + b_{22})$

Từ đây, số lượng phép × giảm xuống 7, phép +/– tăng lên là 18.

Gọi A và B là hai ma trận $n \times n$ ($n = 2^k$):

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Giải thuật

```
Strassen(n, A[1..n][1..n], B[1..n][1..n], C[1..n][1..n]) {
   if (n == 1)
      C = A × B;
   else {
      "Phân chia A thành A11, A12, A21, A22";
      "Phân chia B thành B11, B12, B21, B22";
      Strassen(n/2, A11 + A22, B11 + B22, M1);
      …
      Strassen(n/2, A12 – A22, B21 + B22, M7);
      C11 = M1 + M4 – M5 + M7;
      C12 = M3 + M5;
      C21 = M2 + M4;
      C22 = M1 + M3 – M2 + M6;
      Tổ_hợp(C, C11, C12, C21, C22);
   }
}
```

Đánh giá: Hệ thức truy hồi chia để trị là:

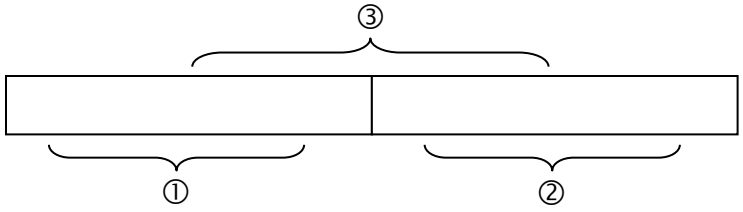
$$T(n) = 7T\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2 \text{ với } T(1) = 1$$

Dựa trên định lý chủ, ta có:

$$T(n) \in \Theta\left(n^{\log_2 7}\right) \approx \Theta(n^{2.81})$$

Bài toán Tìm tổng lớn nhất của dãy con liên tục

Phát biểu: Cho dãy số nguyên *n* phần tử: a_1, a_2, …, a_n. Hãy tìm (và nhận diện dãy con tương ứng) giá trị lớn nhất của $\sum_{k=i}^{j} a_k$ với $1 \le i \le k \le j \le n$. Nếu mọi số nguyên của dãy đều là số âm thì trả về 0.



Giải thuật

```
int sumMax(a[1..n], l, r) {
   if (l == r)
      return   max(a[l], 0);

   c = (l + r) / 2;

   maxLS = sumMax(a, l, c);
   maxRS = sumMax(a, c + 1, r);

   tmp = maxLpartS = 0;
   for (i = c; i ≥ l; i--) {
      tmp += a[i];
      if (tmp > maxLpartS)
         maxLpartS = tmp;
   }
   tmp = maxRpartS = 0;
   for (i = c + 1; i ≤ r; i++) {
      tmp += a[i];
      if (tmp > maxRpartS)
         maxRpartS = tmp;
   }

   tmp = maxLpartS + maxRpartS;
   return   max(tmp, maxLS, maxRS);
}
max = sumMax(a, 1, n);
```

Đánh giá: Hệ thức truy hồi chia để trị là

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \; với \; T(1) = 1$$

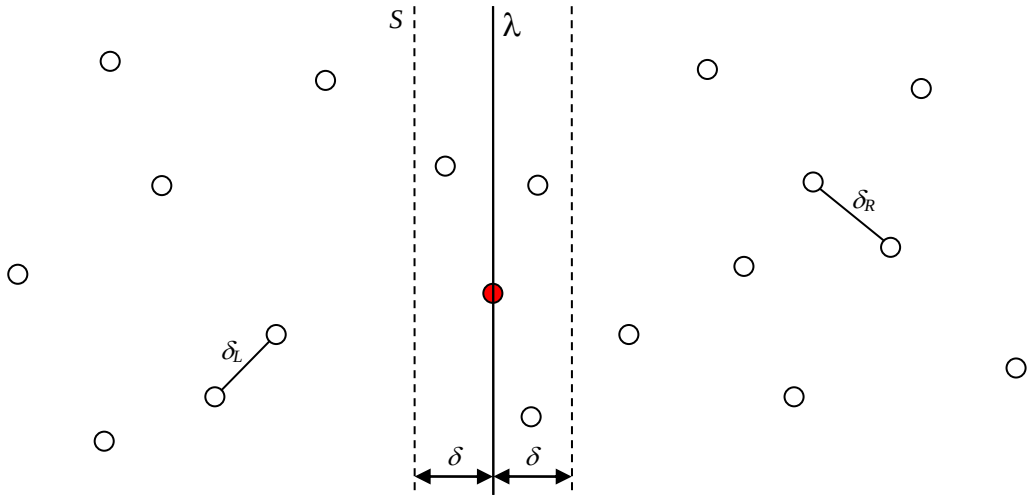
Theo định lý chủ, $T(n) \in \Theta(n \log n)$

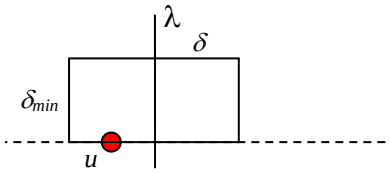
Bài toán cặp (điểm) gần nhất

Phát biểu: Cho tập P gồm n điểm nằm trên mặt phẳng:

$$P = \{p_1, p_2, \ldots, p_n\}$$

Tìm khoảng cách ngắn nhất giữa hai điểm trong tập P.





Giải thuật

```
ClosestPair(Point P[1..n]) {
   Q = P;
   Sắp xếp tập điểm P theo thứ tự hoành độ không giảm;
   Sắp xếp tập điểm Q theo thứ tự tung độ không giảm;
   δ = ClosestPairRec(P, Q);
}

ClosestPairRec(Point P[1..n], Point Q[1..n]) {
   if (|P| ≤ 3)
      Tìm cặp nhỏ nhất theo cách thông thường và trả về;

   λ = P[⌈n/2⌉].x // Giá trị trung vị, theo hoành độ
   Sao chép ⌈n/2⌉ điểm đầu tiên trong P vào P_L;
   Sao chép cũng ⌈n/2⌉ điểm này trong Q vào Q_L, duy trì thứ tự;
   Sao chép ⌊n/2⌋ điểm còn lại trong P vào P_R;
   Sao chép cũng ⌊n/2⌋ điểm này trong Q vào Q_R, duy trì thứ tự;

   δ_L = ClosestPairRec(P_L, Q_L);
   δ_R = ClosestPairRec(P_R, Q_R);
   δ = min(δ_L, δ_R);

   Duyệt tuần tự Q, sao chép các điểm p thỏa |p.x - λ| < δ vào mảng S[1..k];
   δ_min = δ;
   for (i = 1; i < k; i++) {  // S[i] ≡ u
      j = i + 1;              // S[j] ≡ v
      while (j ≤ k) && (|S[i].y - S[j].y| < δ_min) {
         tmp = √((S[i].x − S[j].x)² + (S[i].y − S[j].y)²);
         if (tmp < δ_min)
            δ_min = tmp;
         j++;
      }
   }
   return   δ_min;
}
```

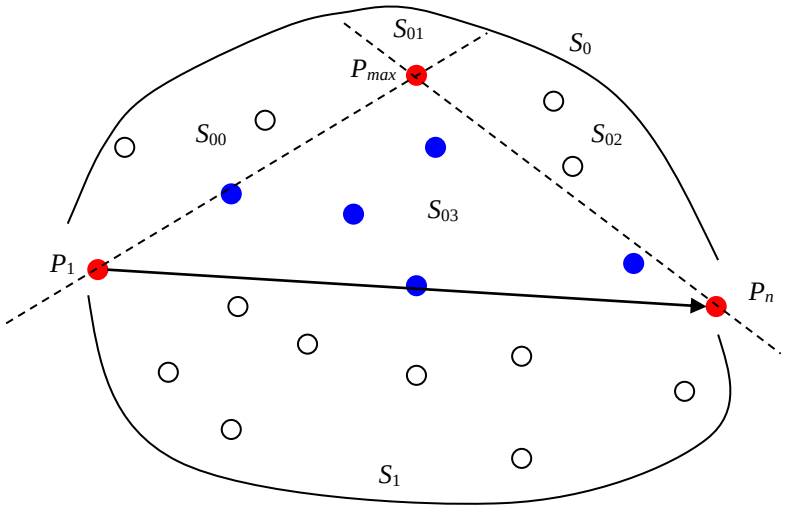
Đánh giá: Hệ thức truy hồi chia để trị là

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \in \Theta(n \log n)$$

Bài toán Bao đóng lồi

Giải thuật có tên gọi là *Quickhull*, vì bước "phân hoạch" gần giống như *Quicksort*.

Gọi $S = \{P_1(x_1, y_1), P_2(x_2, y_2), \ldots, P_n(x_n, y_n)\}$ là tập gồm n điểm trên mặt phẳng hai chiều. *Với mục đích in dãy các điểm cực theo chiều kim đồng hồ, chúng ta sắp xếp lại các điểm trong tập S theo thứ tự tăng dần của hoành độ.*



Cho tọa độ 3 đỉnh $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ và $P_3(x_3, y_3)$.

$$det = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + x_2y_3 + x_3y_1 - x_3y_2 - x_2y_1 - x_1y_3$$

- Diện tích tam giác $\Delta(P_1, P_2, P_3)$ sẽ bằng một nửa của độ lớn của định thức *det*.
- *det* > 0: Điểm $_{P3}$ nằm bên trái vector $\overrightarrow{P_1P_2}$.
- *det* < 0: Điểm P_3 nằm bên phải vector $\overrightarrow{P_1P_2}$.
- *det* = 0: Điểm P_3 nằm trên vector $\overrightarrow{P_1P_2}$.

Giải thuật

```
delete_right(S[1..n], P1, Pn) {
   point leftArea[1..n];

   leftArea[1] = P1;     leftArea[2] = Pn;
   cnt = 2;
   for (mỗi điểm P ∈ S \ {P1, Pn})
      if (ontheLeft(P1, Pn, P))
         leftArea[cnt++] = P;
   return <leftArea, cnt>;
}
point getPmax(point S[1..n]) {
   P1 = S[1];   Pn = S[2];
   maxarea = 0;
   for (mỗi điểm P ∈ S \ {P1, Pn}) {
      area = detVal(P1, Pn, P);
      if (area > maxarea) {
         maxarea = area;
         Pmax = P;
      }
   }
   return Pmax;
}
void qh(S[1..n]) {
   point S00[1..n], S02[1..n];
   if (n == 2)
      return;
   if (n == 3) {
      hull ∪= S[3];
      return;
   }
   P1 = S[1];   Pn = S[2];
   Pmax = getPmax(S);
   hull ∪= Pmax;
   <S00, cnt00> = delete_right(S, P1, Pmax);
   qh(S00[1 .. cnt00]);
   <S02, cnt02>  = delete_right(S, Pmax, Pn);
   qh(S02[1 .. cnt02]);
}
void main() {
   // Giả sử tập điểm S đã sắp theo hoành độ
   hull = S[1] ∪ S[n];
   <S0, cnt0> = delete_right(S, S[1], S[n]);
   <S1, cnt1> = delete_right(S, S[n], S[1]);
   qh(S0[1 .. cnt0]);
   qh(S1[1 .. cnt1]);
   print_hull();
}
```

Bài toán đổi tiền xu

Phát biểu: Giả sử có các mệnh giá tiền xu là $x_1, x_2, \ldots, x_k$. Tìm số lượng đồng tiền xu nhỏ nhất để có thể đổi *n* xu.

Giải thuật

```
moneyChange(coins[1..k], money) {
   minCoins = money;

   for (i = 1; i ≤ k; i++)
      if (coins[i] == money)
         return   1;

   for (i = 1; i ≤ money / 2; i++) {
      tmpSum = moneyChange(coins, i) + moneyChange(coins, money - i);
      if (tmpSum < minCoins)
         minCoins = tmpSum;
   }

   return   minCoins;
}
```

Đánh giá: Hệ thức truy hồi chia để trị là

$$T(n) = \sum_{i=1}^{\lfloor n/2 \rfloor} \big(T(i) + T(n-i)\big) + \Theta\left(\frac{n}{2}\right) \in \Omega(2^n)$$

Giải thuật (cải tiến)

```
moneyChange(coins[1..k], money) {
   minCoins = money;

   for (i = 1; i ≤ k; i++)
      if (coins[i] == money)
         return   1;

   for (i = 1; i ≤ k; i++)
      if (money > coins[i]) {
         tmpSum = 1 + moneyChange(coins, money - coins[i]);
         if (tmpSum < minCoins)
            minCoins = tmpSum;
      }

   return   minCoins;
}
```