Chia để trị (5)

Nguyễn Thanh Bình
Khoa Công nghệ Thông tin
Trường đại học Bách khoa
Đại học Đà Nẵng

Chia để trị (Divide and Conquer)

- Nguyên tắc
  - Nhiều thuật toán có cấu trúc đệ quy
    - Để giải quyết vấn đề đặt ra, thuật toán gọi lại chính nó để giải quyết các vấn đề con có kích thước nhỏ hơn, cuối cùng kết hợp các kết quả thu được giải pháp
  - Gồm các bước
    - **Chia**: chia vấn đề thành các vấn đề con
    - **Trị**: giải quyết các vấn đề con một cách đệ quy, nếu vấn đề con có kích thước đủ nhỏ thì giải quyết trực tiếp
    - **Kết hợp**: các kết quả của các vấn đề con là giải pháp cho vấn đặt ra

Chia để trị

- Cấu trúc chung

```
chia-để-trị(x: bài toán) : giải pháp
begin
   if (x nhỏ và đơn giản) then
           return (giải pháp)
   else
           chia x thành nhiều bài toán con x1, x2, ... xk
           for i from 1 to k do
               si = chia-để-trị(xi)
           endfor
           kết hợp các giải pháp si thành giải pháp s của x
           return (s)
  endif
end
```

Một số ứng dụng

- Tìm giá trị lớn nhất và giá trị nhỏ nhất
- Nhân hai ma trận
- Quicksort
- Chọn phần tử
- Tính bao đóng lồi

Tìm giá trị lớn nhất và giá trị nhỏ nhất

- Bài toán
  - Tìm giá trị lớn nhất (max) và giá trị nhỏ nhất (min) trong một danh sách A[1..n]. Cần sử dụng bao nhiêu phép so sánh giữa các phần tử của A ?
- Thuật toán vét cạn

```
maxmin(A, n)
begin
   max = A[1]
   min = A[1]
   for i from 2 to n
          if (A[i] > max) then max = A[i]
          endif
          if (A[i] < min) then min = A[i]
          endif
   endfor
   return (max, min)
end
```

Tìm giá trị lớn nhất và giá trị nhỏ nhất

- Thuật toán chia để trị

```
maxmin(A,x,y)
begin
   if (y-x ≤ 1) then
       return (max(A[x], A[y])), min(A[x], A[y]))
   else
       (max1, min1) = maxmin(A, x, ⌊(x+y)/2⌋)
       (max2, min2) = maxmin(A, ⌊(x+y)/2⌋+1, y)
       return (max(max1, max2), min(min1, min2))
   endif
end
```

Tìm giá trị lớn nhất và giá trị nhỏ nhất

- Phân tích thuật toán
  - Tính số phép so sánh
    - C(n): số phép so sánh, với n = y – x + 1
    - Giả sử n luỹ thừa của 2, nghĩa là y – x lẽ và x+y cũng lẽ
    - Kích thước các vấn đề con

$$\lfloor (x+y)/2 \rfloor - x + 1 = \frac{x+y-1}{2} - x + 1 = \frac{y-x+1}{2} = \frac{n}{2}$$

$$y - (\lfloor (x+y)/2 \rfloor + 1) + 1 = y - \frac{x+y-1}{2} = \frac{y-x+1}{2} = \frac{n}{2}$$

    - Vậy

$$C(n) = \begin{cases} 1 & if\ n = 2 \\ 2C(n/2) + 2 & else \end{cases}$$

Tìm giá trị lớn nhất và giá trị nhỏ nhất

- Phân tích thuật toán
  - Tính số phép so sánh

$$C(n) = 2C(n/2) + 2$$
$$= 2^2 C(n/4) + 2^2 + 2$$
$$= 2^3 C(n/8) + 2^3 + 2^2 + 2$$
$$= 2^i C(n/2^i) + \sum_{k=1}^{i} 2^k = 2^{\log n - 1} C(2) + \sum_{k=1}^{\log n - 1} 2^k$$
$$= 2^{\log n - 1} + \frac{1 - 2^{\log n - 1 + 1}}{1 - 2} = \frac{2^{\log n}}{2} + 2^{\log n} - 1 = \frac{3}{2} n - 1$$

➡ Thuật toán chia để trị chỉ sử dụng 75% số phép toán so sánh so với thuật toán vét cạn

Nhân hai ma trận

- Bài toán
  - Nhân hai ma trận vuông có nxn phần tử: C = A.B
- Thuật toán vét cạn

```
matrixproduct (A, B, n)
begin
        for i from 1 to n
                for j from 1 to n
                        C(i,j) = 0
                        for k from 1 to n
                                C(i,j) = C(i,j) + A(i,k) * B(k,j)
                        endfor
                endfor
        endfor
        return (C)
end
```

Thuật toán thực hiện $O(n^3)$ phép cộng và phép nhân

Nhân hai ma trận

- Thuật toán chia để trị (1)
  - Giả sử $n = 2^k$
  - Chia các ma trận A, B, C thành các ma trận có kích thước n/2, khi đó C = A.B tương ứng

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

  - Từ đó, ta có

    r = ae + bg

    s = af + bh

    t = ce + dg

    u = cf + dh

Nhân hai ma trận

- Thuật toán chia để trị (2)

```
matrixproduct (A, B, n)
begin
   if (n = 1) then return (A.B)
   else
         Chia A và B thành 8 ma trận kích thước n/2: a, b, ..., h
         r = matrixproduct(a, e, n/2) + matrixproduct(b, g, n/2)
         s = matrixproduct(a, f, n/2) + matrixproduct(b, h, n/2)
         t = matrixproduct(c, e, n/2) + matrixproduct(d, g, n/2)
         u = matrixproduct(c, f, n/2) + matrixproduct(d, h, n/2)
         Ghép r, s, t và u có ma trận C
         return C
   endif
end
```

Độ phức tạp: $C(n) = 8C(n/2) + n^2$, $C(1) = 1$

trong đó, n^2 là số phép cộng

Nhân hai ma trận

- Thuật toán chia để trị (3)

$$
\begin{aligned}
C(n) &= 8C(n/2) + n^2 \\
&= 8(8C(n/4) + (n/2)^2) + n^2 \\
&= 8^2 C(n/4) + 2n^2 + n^2 \\
&= 8^i C(n/2^i) + n^2 \sum_{k=0}^{i-1} 2^k = 8^{\log n} C(1) + n^2 \sum_{k=0}^{\log n - 1} 2^k \\
&= 8^{\log n} + n^2 \frac{1 - 2^{\log n - 1 + 1}}{1 - 2} = 8^{\log n} + n^2 (2^{\log n} - 1) = \\
&= 2^{3\log n} + n^2(n-1) = n^3 + n^2(n-1) = 2n^3 - n^2
\end{aligned}
$$

$\Rightarrow$ $\boxed{O(n^3)}$

Nhân hai ma trận

- Thuật toán Strassen (1)

  $m_1 = (a+c)(e+f)$

  $m_2 = (b+d)(g+h)$

  $m_3 = (a-d)(e+h)$

  $m_4 = a(f-h)$

  $m_5 = (c+d)e$

  $m_6 = (a+b)h$

  $m_7 = d(g-e)$

  - Khi đó

  $r = m_2+m_3-m_6-m_7$

  $s = m_4+m_6$

  $t = m_5+m_7$

  $u = m_1-m_3-m_4-m_5$

Chỉ thực hiện 7 phép nhân ma trận so với 8 phép nhân ma trận của thuật toán chia để trị

Thực hiện nhiều phép cộng và trừ ma trận hơn thuật toán chia để trị

Nhân hai ma trận

- Thuật toán Strassen (2)
  - Tại sao đúng

$r = m_2+m_3-m_6-m_7$

$= (b+d)(g+h) + (a-d)(e+h) - (a+b)h - d(g-e)$

$= bg+bh+dg+dh+ae+ah-de-dh-ah-bh-dg+de$

$= ae + bg$

$s = m_4+m_6 = a(f-h) + (a+b)h = af + bh$

$t = m_5+m_7 = (c+d)e + d(g-e) = ce + dg$

$u = m_1-m_3-m_4-m_5$

$= (a+c)(e+f) - (a-d)(e+h) - a(f-h) - (c+d)e$

$= ae+af+ce+cf-ae-ah+de+dh-af+ah-ce-de$

$= cf + dh$

Nhân hai ma trận

- Thuật toán Strassen (3)

```
matrixproduct (A, B, n)
begin
   if (n = 1) then return (A.B)
   else
          Chia A và B thành 8 ma trận kích thước n/2: a, b, …, h
          m1 = matrixproduct(a+c, e+f, n/2)
          m2 = matrixproduct(b+d, g+h, n/2)
          m3 = matrixproduct(a-d, e+h, n/2)
          m4 = matrixproduct(a, f-h, n/2)
          m5 = matrixproduct(c+d, e, n/2)
          m6 = matrixproduct(a+b, h, n/2)
          m7 = matrixproduct(d, g-e, n/2)
          r = m2 + m3 - m6 - m7
          s = m4 + m6
          t = m5 + m7
          u = m1 - m3 - m4 - m5
          endif
end
```

Nhân hai ma trận

- Thuật toán Strassen (4)
  - Độ phức tạp

$$C(n) = 7C(n/2) + 18n^2/4$$
$$C(1) = 1$$

Nhân hai ma trận

- Thuật toán Strassen (5)
  - Độ phức tạp

$$C(n) = 7C(n/2) + \frac{9}{2}n^2$$

$$= 7(7C(n/4) + \frac{9}{2}(n/2)^2) + \frac{9}{2}n^2$$

$$= 7^2 C(n/8) + \frac{9}{2}7n^2/4 + \frac{9}{2}n^2$$

$$= 7^i C(n/2^i) + \frac{9}{2}n^2 \sum_{k=0}^{i-1}\left(\frac{7}{4}\right)^k = 7^{\log n} C(1) + \frac{9}{2}n^2 \sum_{k=0}^{\log n-1}\left(\frac{7}{4}\right)^k$$

$$= 7^{\log n} + \frac{9}{2}n^2 \frac{(7/4)^{\log n-1+1} - 1}{7/4-1} = 7^{\log n} + \frac{9}{2}n^2 \frac{((7/4)^{\log n} - 1)}{3/4} =$$

$$= n^{\log 7} + 6n^2((7/4)^{\log n} - 1) = n^{\log 7} + 6n^2(n^{\log 7 - \log 4} - 1) = n^{\log 7} + 6n^2\left(\frac{n^{\log 7}}{n^2} - 1\right)$$

$$= O(n^{\log 7}) = O(n^{2.8})$$

Quicksort

- Thuật toán chia để trị

```
quicksort (A)
begin
   if (n = 1) then return (A)
   else
         Chọn một phần tử x trong danh sách A
         Chia danh sách A thành A1, A2, A3 sao cho các phần tử
         của A1 nhỏ hơn x, các phần tử của A2 bằng x và các phần
         tử của A3 lớn hơn x
         return (quicksort(A1), A2, quicksort(A3))
   endif
end
```

Quicksort

- Độ phức tạp trong trường hợp trung bình (1)

  - C(n): số phép so sánh khi sắp xếp danh sách A có n phần tử khác nhau
  - C(0) = C(1) = 0
  - Giả sử x là phần tử nhỏ thứ i trong danh sách A

    $|A_1| = i-1$

    $|A_2| = 1$

    $|A_3| = n-i$
  - Giả sử xác xuất phần tử x nhỏ thứ i trong A là 1/n

Quicksort

- Độ phức tạp trong trường hợp trung bình (2)
  - Lời gọi đệ quy cần thời gian trung bình C(i-1) và C(n-i), với i ∈ [1..n] với xác xuất 1/n
  - Để chia A thành A_1, A_2 và A_3 cần n-1 phép so sánh
  - Vậy, với n ≥ 2

$$C(n) = \frac{1}{n}\sum_{i=1}^{n}\big(C(i-1) + C(n-i)\big) + n - 1$$

  - Mà

$$\sum_{i=1}^{n}\big(C(i-1) + C(n-i)\big) = \sum_{i=1}^{n} C(i-1) + \sum_{i=1}^{n} C(n-i) = \sum_{i=0}^{n-1} C(i) + \sum_{i=0}^{n-1} C(i) = 2\sum_{i=2}^{n-1} C(i)$$

  - Vậy, với n ≥ 2

$$C(n) = \frac{2}{n}\sum_{i=2}^{n-1} C(i) + n - 1 \qquad (1)$$

Quicksort

- Độ phức tạp trong trường hợp trung bình (3)
  - Nhân hai vế của (1) với n, với n ≥ 2 có

$$nC(n) = 2\sum_{i=2}^{n-1} C(i) + n^2 - n \qquad (2)$$

  - Thay n bởi n-1, với n ≥ 3 có

$$(n-1)C(n-1) = 2\sum_{i=2}^{n-2} C(i) + n^2 - 3n + 2 \qquad (3)$$

  - Lấy (2) trừ (3)

$$nC(n) - (n-1)C(n-1) = 2C(n-1) + 2(n-1)$$

  - Vậy

$$nC(n) = (n+1)C(n-1) + 2(n-1) \qquad (4)$$

  - Chia hai vế của (4) cho n(n+1)

$$C(n)/(n+1) = C(n-1)/n + 2(n-1)/n(n+1) \qquad (5)$$

Quicksort

- Độ phức tạp trong trường hợp trung bình (4)
  - Đặt S(n) = C(n)/(n+1), từ (5) ta có, với n≥3

$$S(n) = S(n-1) + 2(n-1)/n(n+1) \qquad (6)$$

với S(0) = S(1) = 0

(5) đúng cả khi n = 2, S(2)=C(2)/3=1/3

  - Vậy

$$S(n) \le \begin{cases} 0 & \text{if } n \le 1 \\ S(n-1) + 2/n & \text{else} \end{cases}$$

$$S(n) \le S(n-1) + 2/n \le S(n-2) + 2/(n-1) + 2/n$$

$$\le S(n-3) + 2/(n-2) + 2/(n-1) + 2/n$$

$$\le S(n-i) + 2\sum_{k=n-i+1}^{n} 1/k$$

Quicksort

- Độ phức tạp trong trường hợp trung bình (5)
  - Thay i = n-1, ta có

$$S(n) \le S(1) + 2\sum_{k=2}^{n}\frac{1}{k} = 2\sum_{k=2}^{n}\frac{1}{k} \le 2\int_{1}^{n}\frac{1}{x}dx = 2\ln n$$

  - Vậy

$$\begin{aligned} C(n) &= (n+1)S(n) \\ &\le 2(n+1)\ln n \\ &\le 2(n+1)\frac{\log n}{\log e} \\ &\le 1.386(n+1)\log n \end{aligned}$$

  - Vậy O(nlog(n))

Quicksort

- Độ phức tạp trong trường hợp xấu nhất
  - Danh sách A đã được sắp xếp và ta chọn x là phần tử đầu tiên

$$C(n) = \begin{cases} 0 & \text{if } n \le 1 \\ C(n-1) + n - 1 & \text{else} \end{cases}$$

  - Vậy: $\Theta(n^2)$

Quicksort

- Độ phức tạp trong trường hợp tốt nhất
  - Phần tử x được chọn luôn là giá trị trung bình, tức là chia thành hai danh sách con có kích thước ~ n/2, nghĩa là i=n/2

$$C(n) = \begin{cases} 0 & \text{if } n \le 1 \\ 2C(n/2) + n - 1 & \text{else} \end{cases}$$

  - Vậy: O(nlog(n))

Quicksort

- Cài đặt

```
quicksort (A, l, r)
begin
   i = l; j = r
   x = một phần tử thuộc A[l..r]
   repeat
           while (A[i]<x) do i = i + 1 endwhile
           while (x>A[j]) do j = j - 1 endwhile
           if (i≤j) then
              hoán đổi A[i] và A[j]
              i = i + 1; j = j - 1
           endif
   until (i>j)
   if (l<j) then quicksort(A,l,j) endif
   if (i<r) then quicksort(A,i,r) endif
end
```

Chọn phần tử

- Bài toán
  - Cho danh sách A có n phần tử khác nhau, chọn phần tử lớn thứ k
  - Nếu k = n/2, vấn đề chọn phần tử trung bình
    - Dùng để chọn chốt trong thuật toán Quicksort

- Giải pháp đơn giản
  - Sắp xếp mảng, sau đó chọn phần tử thứ k
  - Độ phức tạp sắp xếp mảng *O(nlogn)*, độ phức tạp bài toán chọn phần tử cũng là *O(nlogn)*

- Tồn tại giải pháp tốt hơn ?

Chọn phần tử

- Giải pháp tốt hơn (1)
  - Ý tưởng
    - chia danh sách A thành các danh sách con, mỗi danh sách có 5 phần tử,
    - tìm phần tử trung bình của các danh sách con,
    - sau đó tìm phần tử trung bình của các phần tử trung bình
  - Đề xuất bởi M. R. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest and R. E. Tarjan

Chọn phần tử

- Giải pháp tốt hơn (2)

```
F(A, k)
begin
   Chia A thành các danh con có 5 phần tử (danh sách con
    cuối cùng có thể ít hơn 5 phần tử)
   S1 = {các phần tử trung bình từ các danh sách con}, |S1|=m
   x0 = F(S1, m/2) // phần tử trung bình của các phần tử trung bình
   S2 = {x ∈ S| x < x0}, S3 = {x ∈ S| x > x0}
   if (|S2| ≤ k) then return (F(S2, k))
   else if (|S3 ≥ n-k+1|) then return (F(S3, n-k+1))
         else return (x0)
         endif
   endif
end
```

Chọn phần tử

- Giải pháp tốt hơn (3)
  - Thuật toán cho độ phức tạp trong trường hợp trung bình là O(n)
  - Chi tiết hơn
    - M. R. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest and R. E. Tarjan, Time bounds for selection, J. Comput. System Sci. 7 (1972) 448-461

Tính bao đóng lồi

- Bài toán
  - Bao đóng lồi của tập hợp E gồm n điểm trong không gian hai chiều là một đa giác lồi sao cho các đỉnh của đa giác là các điểm thuộc E và tất cả các điểm của E đều nằm bên trong hoặc trên các cạnh của đa giác
  - Xác định bao đóng lồi của tập E gồm n điểm

  - Tính chất
    1. Các đỉnh của đa giác lồi thuộc tập E
    2. Một đường thẳng bất kỳ chia mặt phẳng làm hai phần. Trong mỗi phần, điểm xa đường thẳng nhất trong số n điểm là một đỉnh của đa giác lồi
    3. Một đoạn thẳng nối hai điểm trong số n điểm là một cạnh của đa giác lồi, nếu tất cả các điểm còn lại nằm về một phía của đoạn thẳng

Tính bao đóng lồi

- Tồn tại nhiều thuật toán xác định bao đóng lồi
  - Trong đó có thuật toán chia để trị đề xuất bởi F. Preparata & S.J. Hong
- Ý tưởng
  - Giả sử P và Q là hai điểm của bao đóng lồi (chẳng hạn hai điểm có hoành độ lớn nhất và nhỏ nhất)
  - Đường thẳng PQ chia E thành hai phần E_0 và E_0'
  - Theo tính chất 3, PQ là cạnh của bao lồi E_0 và E_0'
  - Một khi có được bao đóng lồi của E_0 và E_0' thì hợp chúng lại sẽ có được bao đóng lồi của E
  - Xét E_0, giả sử điểm xa PQ nhất là S, theo tính chất 2, S thuộc bao đóng lồi của E_0
  - Chia E_0 thành E_1 và E_1' :

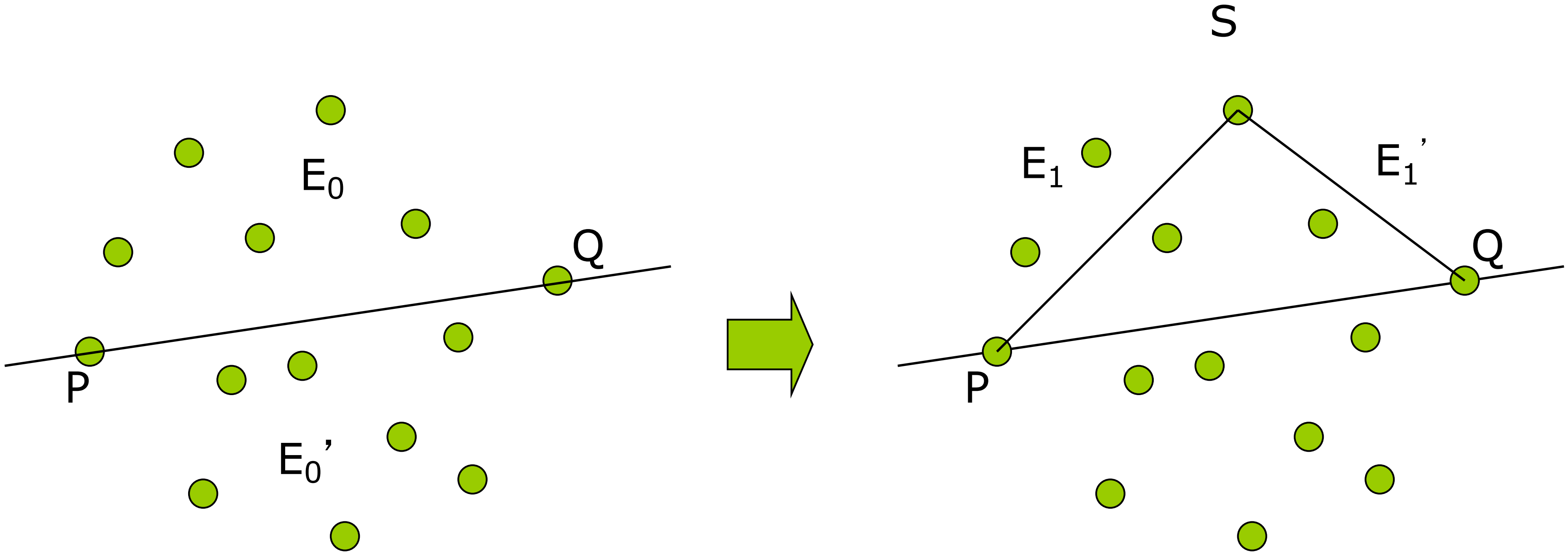
    - E_1 là phần giới hạn bởi PS không chứa Q

    - E_1' là phần giới hạn bởi QS không chứa P

    - các điểm thuộc tam giác PQS không thuộc bao lồi E_0
  - Tiếp tục áp dụng một cách đệ quy cho E_1 và E_1'

Tính bao đóng lồi

- Minh hoạ



Tính bao đóng lồi

- Thuật toán (1)

```
bao-dong-loi (E)
begin
    Tính P và Q
    if (hoành độ P = hoành độ Q) then
        // tất cả n điểm trên đường thẳng
        return (danh sách các điểm của E)
    else
        Tính E0 và E0'
        return (hoa-nhap(nua-baoloi(E0, P, Q), nua-baoloi(E0', Q, P)))
    endif
end
```

Tính bao đóng lồi

- Thuật toán (2)

```
nua-baoloi (E, P, Q)
begin
    Tính S // điểm xa PQ nhất
    if (S trên PQ) then return (PQ) // E rỗng
    else
        Tính E1 = là phần giới hạn bởi PS không chứa Q
        Tính E1' = là phần giới hạn bởi QS không chứa P
        return (hoa-nhap(nua-baoloi(E1, P, S), nua-baoloi(E1', S, Q)))
    endif
end
```

Tính bao đóng lồi

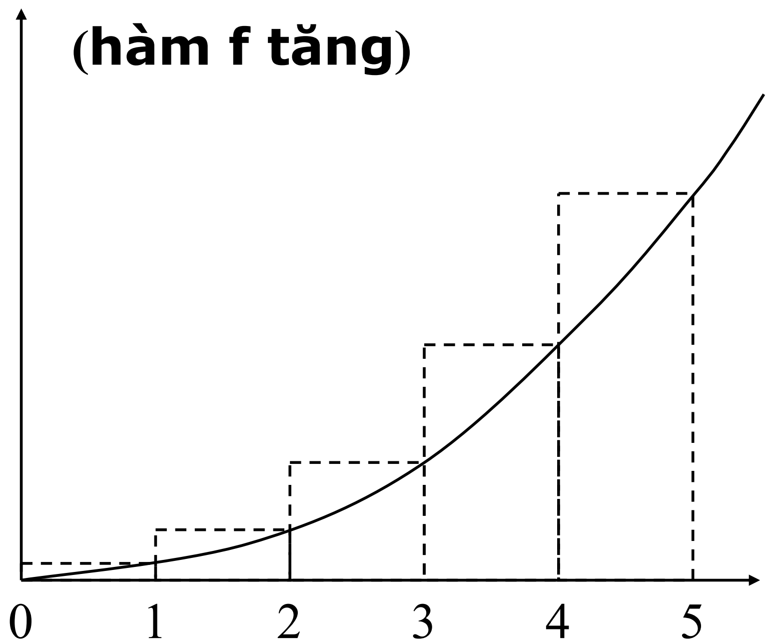
- Độ phức tạp của thuật toán O(*n* log *n*)
- Chi tiết hơn
  - Franco Preparata & S.J. Hong, "Convex Hulls of Finite Sets of Points in Two and Three Dimensions", Comm. ACM 20, 87-93 (1977)

Tính tổng

- Tính tổng (1)

$$F(n)=\sum_{i=1}^{n} f(i)$$

Ví dụ: f(x)=i^p



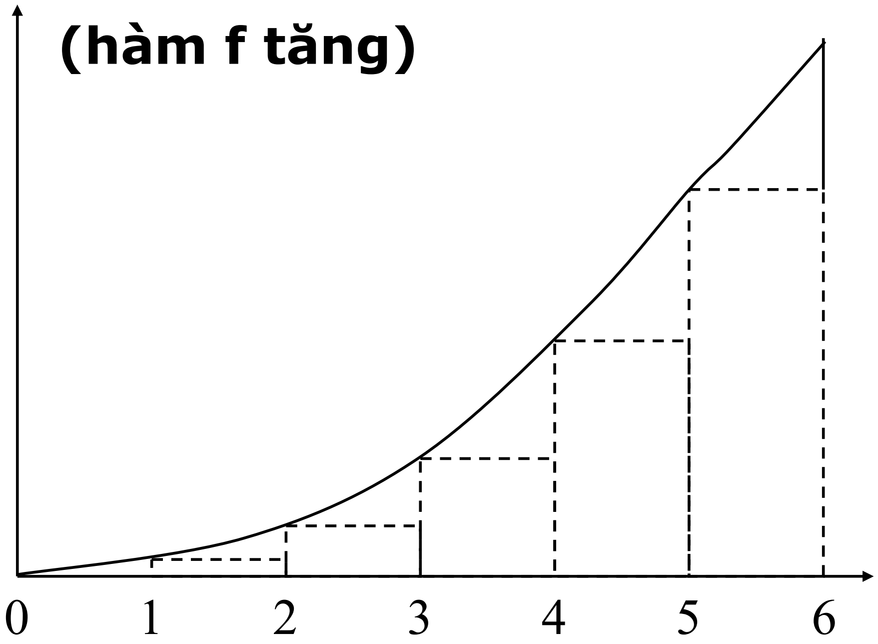
$$\int_0^5 f(t)dt \le \sum_{i=1}^{5} f(i)$$

Tổng quát

$$\int_0^n f(t)dt \le \sum_{i=1}^{n} f(i)$$

Tính tổng

- Tính tổng (2)



$$\sum_{i=1}^{5} f(i) \leq \int_{1}^{6} f(t)dt$$

Tổng quát

$$\sum_{i=1}^{n} f(i) \leq \int_{1}^{n+1} f(t)dt$$

Vậy:

$$\int_{0}^{n} f(t)dt \leq \sum_{i=1}^{n} f(i) \leq \int_{1}^{n+1} f(t)dt$$

Nếu hàm f giảm:

$$\int_{1}^{n+1} f(t)dt \leq \sum_{i=1}^{n} f(i) \leq \int_{0}^{n} f(t)dt$$