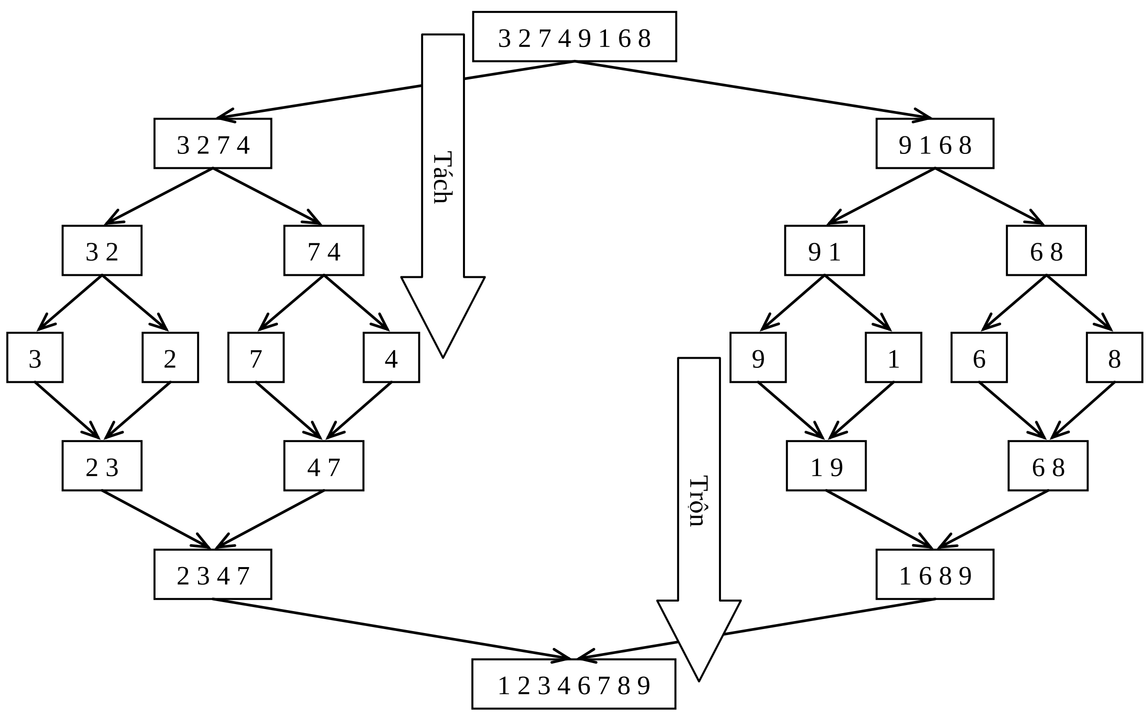
Chương 5: Kỹ thuật Chia để trị (Devide-and-Conquer)

Định lý chủ (*Master theorem*) Cho hệ thức truy hồi chia để trị $T(n) = aT(n/b) + f(n)$. Nếu $f(n) \in \Theta(n^d)$ với $d \geq 0$ thì:

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

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

Sắp xếp trộn (Mergesort)



Giải thuật

```
Mergesort(A[0 .. n - 1]) {
   if (n > 0) {
      Sao chép A[0 .. ⌊n / 2⌋ - 1] sang B[0 .. ⌊n / 2⌋ - 1];
      Sao chép A[⌊n / 2⌋ .. n - 1] sang C[0 .. ⌈n / 2⌉ - 1];
      Mergesort(B[0 .. ⌊n / 2⌋ - 1]);
      Mergesort(C[0 .. ⌈n / 2⌉ - 1]);
      Merge(B, C, A);
   }
}
Merge(B[0 .. p - 1], C[0 .. q - 1], A[r .. r + p + q - 1]) {
   i = j = 0; k = r;
   while (i < p) && (j < q) {
      if (B[i] ≤ C[j]) { A[k] = B[i]; i++; }
      else             { A[k] = C[j]; j++; }
      k++;
   }
   if (i == p) Sao chép C[j .. q - 1] sang A[k .. r + p + q - 1];
   else        Sao chép B[i .. p - 1] sang A[k .. r + p + q - 1];
}
```

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]);    // Phục hồi lại phần hoán vị thừa khi i ≥ j
   swap(a[left], a[j]);
   return   j;
}
```

Nhân số lớn

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 == 1)
         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);
      }
}
```

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 == 1)
         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;
      }
}
```

Nhân ma trận (của) Strassen

Nhân hai ma trận kích thước 2 × 2:

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$
$$= \begin{bmatrix} a_{00} \times b_{00} + a_{01} \times b_{10} & a_{00} \times b_{01} + a_{01} \times b_{11} \\ a_{10} \times b_{00} + a_{11} \times b_{10} & a_{10} \times b_{01} + a_{11} \times b_{11} \end{bmatrix}$$

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

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} m_0 + m_3 - m_4 + m_6 & m_2 + m_4 \\ m_1 + m_3 & m_0 + m_2 - m_1 + m_5 \end{bmatrix}$$

với:

$m_0 = (a_{00} + a_{11}) \times (b_{00} + b_{11})$

$m_1 = (a_{10} + a_{11}) \times b_{00}$

$m_2 = a_{00} \times (b_{01} - b_{11})$

$m_3 = a_{11} \times (b_{10} - b_{00})$

$m_4 = (a_{00} + a_{01}) \times b_{11}$

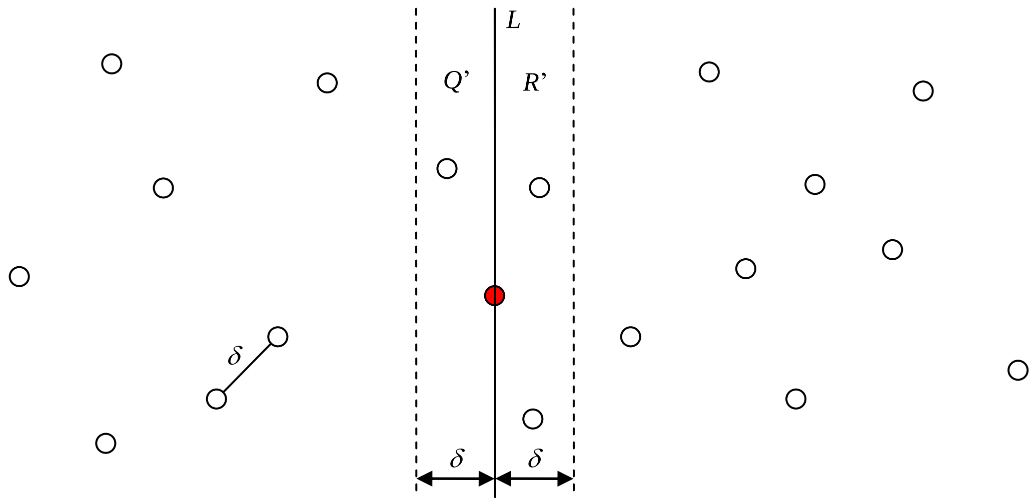
$m_5 = (a_{10} - a_{00}) \times (b_{00} + b_{01})$

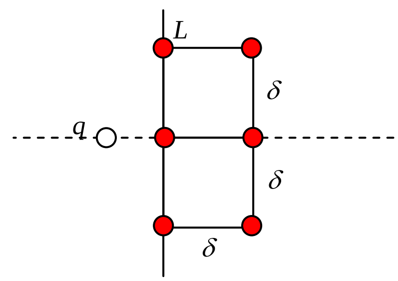
$m_6 = (a_{01} - a_{11}) \times (b_{10} + b_{11})$

Giải thuật

```
Strassen(n, A[0..n-1][0..n-1], B[0..n-1][0..n-1], C[0..n-1][0..n-1]) {
   if (n == 1)
      C = A × B;  // Nhân hai ma trận bậc (1 × 1)
   else {
      "Phân chia A thành A00, A01, A10, A11";
      "Phân chia B thành B00, B01, B10, B11";
      strassen(n/2, A00 + A11, B00 + B11, M0);
      …
      strassen(n/2, A01 – A11, B10 + B11, M6);
      C00 = M0 + M3 – M4 + M6;
      C01 = M2 + M4;
      C10 = M1 + M3;
      C11 = M0 + M2 – M1 + M5;
      Tổ_hợp(C, C00, C01, C10, C11);
   }
}
```

Bài toán cặp (điểm) gần nhất





Qui ước:

- P_x, P_y là tập hợp các điểm thuộc tập P nhưng được sắp theo hoành và tung độ.
- Q_x, Q_y là tập hợp các điểm thuộc tập Q nhưng được sắp theo hoành và tung độ.
- R_x, R_y là tập hợp các điểm thuộc tập Q nhưng được sắp theo hoành và tung độ.

Giải thuật

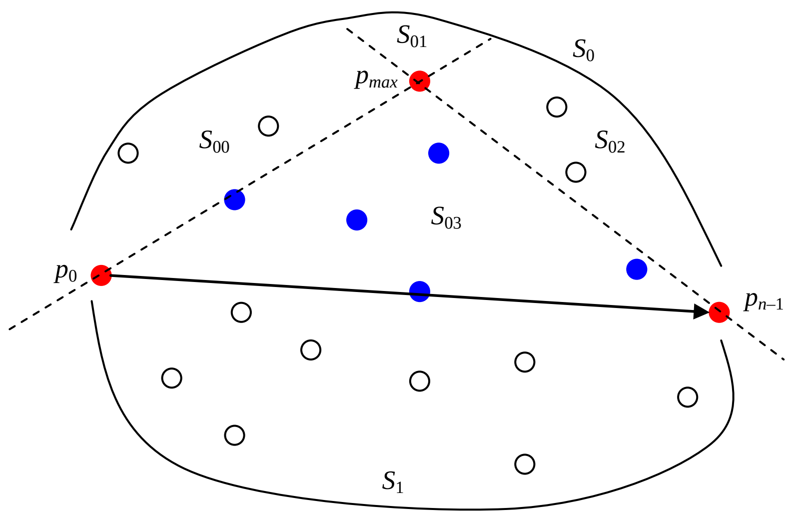
```
ClosestPair(P: Set of Points) {
   Xây dựng lần lượt Px và Py, sử dụng sắp xếp chi phí O(nlogn);
   (p*0,p*1) = ClosestPairRec(Px, Py);
}
ClosestPairRec(Px, Py: Set of Points) {
   if (|P| <= 3)
      Tìm cặp nhỏ nhất theo cách thông thường;
   Duyệt tuần tự Px, xây dựng Qx, Rx;
   Duyệt tuần tự Py, xây dựng Qy, Ry;

   (q*0,q*1) = ClosestPairRec(Qx, Qy);
   (r*0,r*1) = ClosestPairRec(Rx, Ry);
   δ = min(d(q*0,q*1), d(r*0,r*1));
   Duyệt tuần tự Qy, dựa trên δ, xây dựng Q'y;
   Duyệt tuần tự Ry, dựa trên δ, xây dựng R'y;
   Xây dựng Sy bằng cách trộn Q'y và R'y;
```

```
   for (k ∈ Sy)
      Tính khoảng cách từ k đến 6 điểm kế tiếp (gọi chung là điểm h);

   Gọi k và h là hai điểm cho ra khoảng cách ngắn nhất;
   if (d(k, h) < δ)
      return   (k, h);
   else
      if d(q0*, q1*) < d(r0*, r1*)
         return   (q0*, q1*);
      else
         return   (r0*, r1*);
}
```

Bài toán Bao đóng lồi



Giải thuật

```
delete_right(S[0 .. n - 1], p0, p1) {
   point leftArea[0 .. n];

   leftArea[0] = p0;
   leftArea[1] = p1;
   cnt = 2;
   for (mỗi điểm p ∈ S \ {p0, p1})
      if (ontheLeft(p0, p1, p))
         leftArea[cnt++] = p;
   return <leftArea, cnt>;
}
point getPmax(point S[0 .. n - 1]) {
   p0 = S[0];
   p1 = S[1];
   int maxarea = 0;
   for (mỗi điểm p ∈ S \ {p0, p1}) {
      area = detVal(p0, p1, p);
```

```
      if (area > maxarea) {
         maxarea = area;
         Pmax = p;
      }
   }
   return Pmax;
}

void qh(point S[0 .. n - 1]) {
   point S00[0 .. n - 1], S02[0 .. n - 1];

   if (n == 2)
      return;
   if (n == 3) {
      hull ∪= S[2];
      return;
   }
   p0 = S[0];
   p1 = S[1];
   Pmax = getPmax(S);
   hull ∪= Pmax;
   <S00, cont00> = delete_right(S, p0, Pmax);
   qh(S00, cnt00);

   <S02, cont02>  = delete_right(S, Pmax, p1);
   qh(S02, cnt02);
}
void main() {
   // Giả sử tập điểm P đã sắp theo hoành độ
   hull = P[0] ∪ P[n - 1];
   <S0, cnt0> = delete_right(P, P[0], P[n - 1]);
   <S1, cnt1> = delete_right(P, P[n - 1], P[0]);
   qh(S0[0 .. cnt0 - 1]);
   qh(S1[0 .. cnt1 - 1]);
   print_hull();
}
```