Chương 3: Kỹ thuật "Brute Force"

Giới thiệu chung

Chiến lược thiết kế giải thuật đơn giản nhất. Nói chung là kém hiệu quả.

Ví dụ: Tính x^n. Đơn giản là chỉ việc thực hiện dãy n phép nhân: $1 \times x \times x \times \ldots \times x$.

Tuy nhiên, tồn tại một số lý do khiến người ta vẫn cần đến kỹ thuật thiết kế giải thuật này.

Ví dụ: Sắp xếp nổi bọt

```
Bubblesort(a[1 .. n]) {
   for (i = 2; i ≤ n; i++)
      for (j = n; j ≥ i; j--)
         if (a[j - 1] > a[j])
            a[j - 1] ⇆ a[j];
}
```

Ví dụ: Tìm kiếm chuỗi con

```
SequentialStringSearch(T[1 .. n], P[1 .. m]) {
   for (i = 1; i ≤ n − m + 1; i++) {
      j = 0;
      while (j < m) && (P[1 + j] == T[i + j])
         j++;
      if (j == m)
         return   i;
   }
   return -1;
}
```

Bài toán Tìm tổng (của) dãy con liên tục (có giá trị) lớn nhất (The Maximum Contiguous Subsequence Sum Problem)

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

Giải thuật

```
SubsequenceSum(a[1..n]) {
   maxSum = 0;
   for (i = 1; i ≤ n; i++)
      for (j = i; j ≤ n; j++) {
         tmpSum = 0;
         for (k = i; k ≤ j; k++)
            tmpSum += a[k];
         if (tmpSum > maxSum) {
            maxSum = tmpSum;
            left = i;
            right = j;
         }
      }
   return <maxSum, left, right>;
}
```

Đánh giá:

$$A(n) \in \Theta(n^3)$$

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.

Giả định rằng, mệnh giá tiền xu nhỏ nhất là 1 xu để đảm bảo cho bài toán có lời giải.

Tư tưởng của giải thuật *brute-force* là tìm tất cả các khả năng $c_1, c_2, \ldots, c_k$ sao cho:

$$c_1 \times x_1 + c_2 \times x_2 + \cdots + c_k \times x_k = n \text{ và } \sum_{i=1}^{k} c_i \text{ nhỏ nhất}$$

với c_i là số lượng đồng xu có mệnh giá x_i.

Bài toán Cặp (điểm) gần nhất

Phát biểu: Tìm cặp điểm gần nhất trên mặt phẳng tọa độ hình chữ nhật theo độ đo Euclid.

Giải thuật

```
BruteForceClosestPoints(P) {
   dmin = ∞;
   for (i = 1; i ≤ n - 1; i++)
      for (j = i + 1; j ≤ n; j++) {
         d = sqrt((xi - xj)² + (yi - yj)²);
         if (d < dmin) {
            dmin = d;
            point1 = i;
            point2 = j;
         }
      }
   return   <point1, point2>;
}
```

Bài toán Bao đóng lồi

Bài toán tìm bao đóng lồi của tập điểm S chính là tìm đa giác lồi với các đỉnh của đa giác cũng thuộc về S.

Ý tưởng:

"Đoạn thẳng nối hai điểm P và Q của tập điểm S là cạnh của đa giác lồi nếu và chỉ nếu tất cả các điểm khác của S cùng nằm trên một phía của mặt phẳng được phân cách bởi đường thẳng đi qua P và Q".

Hình học giải tích chỉ ra:

- Đường thẳng đi qua hai điểm (x_1, y_1) và (x_2, y_2) được xác định bằng công thức:

$$ax + by = c$$

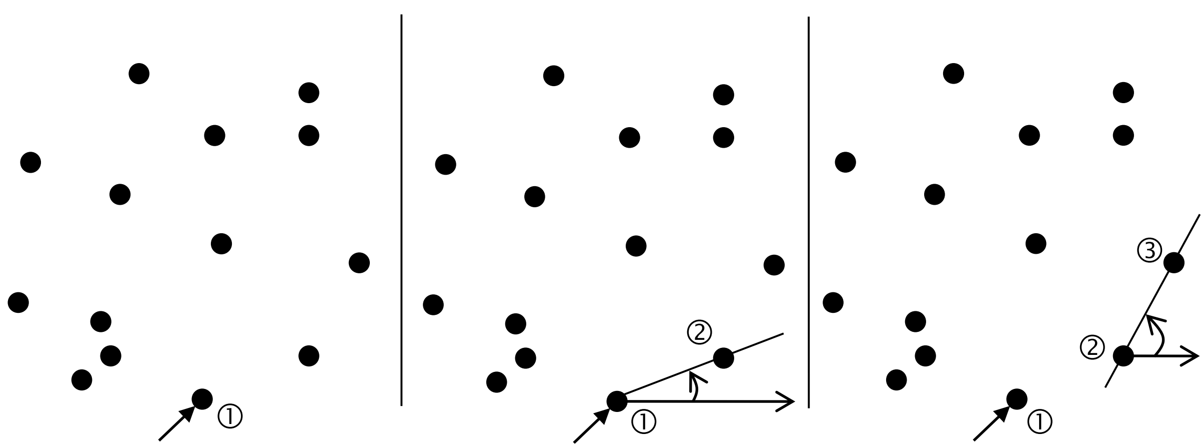
với $a = y_2 - y_1$, $b = x_1 - x_2$, $c = x_1y_2 - x_2y_1$.

- Đoạn thẳng $ax + by = c$ chia mặt phẳng thành hai nửa: Mọi điểm trên một nửa mặt phẳng sẽ khiến cho $ax + by > c$ và mọi điểm trên nửa kia sẽ khiến cho $ax + by < c$. Tọa độ những điểm nằm trên đường thẳng này thỏa $ax + by = c$.

Giải thuật

```
for (Tất cả điểm pᵢ trong tập S: i = 1 → n - 1)
   for (Tất cả điểm qⱼ trong tập S: j = i + 1 → n) {
      Xây dựng đường thẳng pᵢqⱼ;
      if (Tất cả các điểm khác trong S nằm về một phía của đường pᵢqⱼ)
         Bổ sung đoạn pᵢqⱼ vào danh sách kết quả
   }
```

Mở rộng:



Giải thuật

```
computeAngle(point from, point to) {
   angle = atan2(to.y - from.y, to.x - from.x);
   if (angle < 0)
      angle += 2 * π;
   return angle;
}
findNextExtremePoint(S, cur, curAngle) {
   minAngle = 2 * π;
   S \= cur;
   for (Mỗi p trong S) {
      angle = computeAngle(cur, p);
      if (angle < minAngle && angle ≥ curAngle) {
         next = p;
         minAngle = angle;
      }
   }
   S ∪= cur;
   return [next, minAngle];
}
computeConvexHull(S) {
   convexHull = ∅;
   Gọi first là điểm có tung độ nhỏ nhất trong S;
   convexHull ∪= first; // Cần đảm bảo thứ tự (thêm vào cuối danh sách)
   curAngle = 0;
   point cur = first;
   while (true) {
      [next, curAngle] = findNextExtremePoint(S, cur, curAngle);
      if (first == next)
         break;
      convexHull ∪= next;
      cur = next;
   }
   return convexHull;
}
```

Tìm kiếm vét cạn (Exhaustive Search)

Giải thuật tạo tập hợp của các tập con từ tập có kích thước n

```
for (k = 0; k < 2^n; k++)
   In chuỗi bit chiều dài n biểu diễn k;
```

Giải thuật tạo hoán vị

```
Taohoanvi(pivot, a[1 .. n], n) {
   if (pivot == n)
      inhoanvi(a, n);
   else
      for (i = pivot; i ≤ n; i++) {
         a[pivot] ⇆ a[i];
         Taohoanvi(pivot + 1, a, n);
         a[pivot] ⇆ a[i];
      }
}
Taohoanvi(1, a, n);
```

Đánh giá:

$$T(n) = nT(n-1) + \Theta(n) \text{ và } T(1) = 1$$

Khai triển $T(n)$, ta có $T(n) \in \Omega(n!)$

Bài toán đường đi người bán hàng

Phát biểu: Cho *n* thành phố. Tìm con đường *ngắn nhất* trong số các con đường đi qua mọi thành phố (duy nhất một lần) và quay trở về thành phố xuất phát.

Chu trình Hamilton: Cho đồ thị có *n* đỉnh. Một chu trình Hamilton là dãy của *n* + 1 đỉnh kề nhau: v_{i_0}, v_{i_1}, …, $v_{i_{n-1}}$, v_{i_0}, với đỉnh đầu và đỉnh cuối của dãy là một, các đỉnh còn lại khác nhau từng đôi một.

Ý tưởng

- Chọn đỉnh xuất phát v_{i_0}. Xây dựng tất cả các hoán vị của *n* – 1 đỉnh còn lại (trung gian): v_{i_1}, …, $v_{i_{n-1}}$.
- Với mỗi hoán vị, bổ sung đỉnh v_{i_0} vào đầu và cuối rồi kiểm tra xem từng cặp đỉnh liên tiếp có phải là đỉnh kề?
- Nếu tất cả các cặp đỉnh trong hoán vị hiện tại đều là đỉnh kề thì đây là một chu trình Hamilton. Có thể tính chiều dài để xử lý.
- Ngược lại thì chuyển sang hoán vị kế.

Cách tiếp cận này rõ ràng thuộc về nhóm có bậc tăng trưởng Θ(*n*!).

Bài toán túi xách 0-1 (0-1 Knapsack Problem)

Phát biểu: Một chiếc túi có khả năng chứa khối lượng tối đa W. Có n vật với khối lượng $w_1, w_2, \ldots, w_n$ và giá trị tương ứng là $v_1, v_2, \ldots, v_n$. Tìm tập con có giá trị nhất mà chiếc túi có khả năng mang được.

Bài toán có thể được phát biểu lại dưới dạng công thức như sau:

$$maximize \sum_{i=1}^{n} v_i x_i$$

$$subject\ to \sum_{i=1}^{n} w_i x_i \leq W$$

$$x_i = \{0, 1\}, i = 1, .., n$$

Bài toán kết nối (Assignment Problem)

Phát biểu: Có n người và n việc đang tìm nhau. Biết rằng, người thứ i khi có việc thứ j sẽ nhận mức lương C(i, j). Vấn đề là tìm cách giao việc thế nào để tổng chi phí là thấp nhất.

Ví dụ: Bảng C dưới đây chỉ ra các thông tin liên quan

	Việc 1	Việc 2	Việc 3	Việc 4
Người 1	9	2	7	8
Người 2	6	4	3	7
Người 3	5	8	1	8
Người 4	7	6	9	4

Ý tưởng: Phát sinh mọi hoán vị có thể của dãy n số từ 1 đến n. Xem như đây là một dãy Q rồi tính tổng chi phí và xác định hoán vị nào có chi phí thấp nhất.

Chi phí của giải thuật thuộc vào nhóm $\Theta(n!)$.