TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG HÀ NỘI

KHOA TOÁN – CƠ – TIN HỌC

_____o0o_____

BÀI TIỂU LUẬN GIỮA KỲ

MÔN THIẾT KẾ VÀ ĐÁNH GIÁ THUẬT TOÁN

Chủ đề 7: Phương pháp nhánh cận

❖ *Giảng viên:* TS.Nguyễn Thị Hồng Minh

❖ *Nhóm sinh viên thực hiện:*

1. Hoàng Thị Quỳnh Hoa
2. Nguyễn Thị Huyên
3. Đỗ Thị Duyên
4. Dư Thị Lan Hương

❖ *Lớp:* K54A2

Hà Nội – 2012

MỤC LỤC

TÀI LIỆU THAM KHẢO

1. Slide bài giảng của cô + tham khảo tài liệu của các anh chị khóa trên.
2. *http://fit.vimaru.edu.vn/tai_lieu/chi_tiet/2/18/bai_giang_phan_tich_thiet_ke_va_danh_gia_thuat_toan.html*
3. Lê Minh Hoàng, *Giải thuật và lập trình,* Đại học Sư phạm Hà Nội, 1999-2002.

I. Giới thiệu chung về lớp phương pháp thử sai

1. Phân loại

Lớp phương pháp thử sai gồm 3 phương pháp: vét cạn, quay lui và nhánh cận.

1.1. Phương pháp vét cạn (Exhautive)

Duyệt tất cả các phương án tồn tại nghiệm của bài toán để xác định nghiệm đúng.

Nguyên lý Edison: "Tìm kim trong đống rơm!"

Ví dụ: Tìm kiếm phần tử trong dãy, xây dựng chu trình Euler, Hamilton của đồ thị, xếp balo,…

1.2. Phương pháp quay lui (Back tracking)

Theo nguyên tắc vét cạn, nhưng chỉ xét những trường hợp khả quan.

Ví dụ: Bài toán người du lịch, tìm đường trong mê cung, xếp balo,…

1.3. Phương pháp nhánh cận (Branch and Bound)

Thuật toán tìm lời giải cho các bài toán tối ưu dạng liệt kê cấu hình dựa trên nguyên lý đánh giá nhánh cận.

Ví dụ: Bài toán người du lịch, bài toán dãy ABC, bài toán xếp balo,…

=> Vét cạn là cơ sở cho hai phương pháp quay lui và nhánh cận. Cả vét cạn, quay lui và nhánh cận đều thực hiện duyệt cấu trúc cây biểu diễn không gian ứng viên. Điểm phân biệt giữa vét cạn, quay lui và nhánh cận là trong quá trình duyệt (theo chiều sâu) cây, phương pháp quay lui sử dụng một *hàm điều kiện*, tại mỗi nút nếu hàm điều kiện không thỏa mãn, toàn bộ cây con có gốc tại nút hiện tại được bỏ qua; phương pháp nhánh cận sử dụng một *hàm tính cận* để tính trước điểm tối đa (cận) có thể đạt được đối với các ứng viên thuộc cây con có gốc tại nút hiện tại, nếu cận này không cao hơn điểm của ứng viên đã biết thì toàn bộ cây con có gốc tại nút hiện tại được bỏ qua. Có thể tóm tắt hai phương pháp quay lui và nhánh cận theo các công thức:

Vét cạn + Hàm điều kiện = Quay lui,

Vét cạn + Hàm tính cận = Nhánh cận.

2. *Ưu điểm*

Luôn đảm bảo tìm ra nghiệm chính xác. Ngoài ra còn có ưu điểm là tốn ít bộ nhớ và cài đặt đơn giản.

3. *Hạn chế*

Thời gian thực thi rất lớn, độ phức tạp thường ở hàm mũ, do đó chỉ áp dụng với các bài toán nhỏ.

II. Đặt vấn đề

Một trong những bài toán đặt ra trong thực tế là tìm một nghiệm của bài toán thỏa mãn một số điều kiện nào đó và nghiệm đó là tốt nhất theo một tiêu chí cụ thể, nghiên cứu lời giải các bài toán tối ưu thuộc về lĩnh vực quy hoạch toán học, mô hình được sử dụng để tìm kiếm là mô hình cây phân cấp. Việc tìm nghiệm của bài toán thường phải dựa vào việc liệt kê toàn bộ các cấu hình có thể và đánh giá tìm ra cấu hình tốt nhất.

Ví dụ: Bài toán người du lịch có không gian trạng thái là n!

Bài toán K – median có không gian trạng thái là $C_n^k = \frac{n!}{k!(n-k)!}$

Khi đó, không gian tìm kiếm là rất lớn trong khi nhiều trường hợp có thể loại bỏ được ngay. Điều này gây nên tình trạng lãng phí bộ nhớ và mất rất nhiều thời gian. Vấn đề đặt ra là trong quá trình liệt kê lời giải cần tận dụng những thông tin đã có để loại bỏ sớm những phương án chắc chắn không tối ưu. Thuật toán nhánh cận được sử dụng để khắc phục những vấn đề trên.

Hai hướng tiếp cận tìm lời giải tối ưu cho bài toán:

- Tìm từng lời giải, khi hoàn tất một lời giải thì so sánh của nó với chi phí tốt nhất hiện có. Nếu tốt hơn thì cập nhật chi phí tốt nhất mới.
- Với mỗi lời giải, khi xây dựng các thành phần nghiệm luôn kiểm tra điều kiện nếu đi tiếp theo hướng này thì có khả năng nhận được lời giải tốt hơn lời giải hiện có không? Nếu không thì thôi không đi theo hướng này nữa. => Nguyên lý ***nhánh cận*** (*Branch and Bound*).

Nhận dạng những bài toán có thể giải bằng phương pháp nhánh cận: Các bài toán quy hoạch tuyến tính, bài toán tối ưu có nhiều phương án, đòi hỏi cần tìm một phương án khả dĩ nhất hay một phương án tốt nhất.

III. Phương pháp nhánh cận

Trong các phương pháp giải bài toán quy hoạch nguyên, phương pháp nhánh cận là một trong các phương pháp có hiệu quả. Phương pháp nhánh cận được Land A.H và Doig A.G xây dựng năm 1960 giải bài toán quy hoạch nguyên (trình bày Tiết 2), đến 1963 được Little J.D, Murty K.G, Sweeney D.W và Karen C sử dụng thành công giải bài toán người du lịch. Năm 1979, Giáo sư Hoàng Tụy đã ứng dụng thành công phương pháp này vào giải bài toán quy hoạch lõm. Đây là một trong những phương pháp giải các bài toán liệt kê cấu hình có điều kiện tối ưu.

1. Định nghĩa

Nhánh cận là chèn thêm vào đệ quy một bước kiểm tra xem nếu đi nữa thì có thể có được kết quả hay không, nếu không thì quay lui ngay, không xét nữa (vì xét nữa cũng không có kết quả mong muốn).

2. Ý tưởng

Nhánh cận (Branch and Bound): Thuật toán tìm lời giải cho các bài toán tối ưu dạng liệt kê cấu hình dựa trên nguyên lí đánh giá nhánh cận.

3. Nguyên lý đánh giá nhánh cận

Sử dụng các thông tin đã tìm được trong lời giải của bài toán để loại bỏ sớm phương án không dẫn tới lời giải tối ưu.

Một số chú ý khi chọn cận:

- Cận phải đánh giá chính xác tình trạng (giá) của cấu hình (phương án) hiện tại. Nếu quá lỏng thì số cấu hình loại bỏ không đáng kể, nếu quá chặt thì sẽ dẫn tới bỏ sót nghiệm.
- Cận phải tính toán đơn giản. Vì thao tác tính cận thực hiện tại tất cả các bước nên nếu tính cận quá phức tạp thì thời gian rút ngắn nhờ đặt cận tiết kiệm được thì lại mất đáng kể cho việc tính cận.
  VD: *Cải tiến cận* với bài toán người du lịch (được phân tích ở mục *III.8.1.3*)

4. Bản chất

- Sử dụng phương pháp quay lui nhưng tại mỗi bước đưa thêm thao tác đánh giá giá trị phương án hiện có.
- Nếu đó là phương án tối ưu hoặc có hy vọng trở thành phương án tối ưu (tức là tốt hơn phương án hiện có) thì cập nhật lại phương án tối ưu hoặc đi tiếp theo hướng đó.
- Trong trường hợp ngược lại thì bỏ qua hướng đang xét.

5. Mô hình

Không gian của bài toán (tập khả năng) $D = \{(x_1, x_2,\ldots, x_n)\}$ gồm các cấu hình liệt kê có dạng $(x_1, x_{2,\ldots\ldots,} x_n)$.

Mỗi cấu hình x sẽ xác định một giá trị hàm chi phí $f(x)$.

$f : D \to Z \quad f(x) = C \ (C \in Z)$

Nghiệm của bài toán: $x = (x_1, x_2,\ldots, x_n)$ sao cho $f(x)$= *giá trị tối ưu (max/min)*.

- Cho x_i nhận lần lượt các giá trị có thể. Với mỗi giá trị thử gán cho x_i xét khả năng chọn $x_{i+1}, x_{i+2}\ldots$
- Tại mỗi bước i: Xây dựng thành phần x_i
  - Xác định x_i theo khả năng *v*.
  - Tính chi phí lời giải nhận được. Nếu "*tốt hơn*" lời giải hiện thời thì chấp nhận x_i theo khả năng *v*. Tiếp tục xác định $x_{i+1},\ldots$đến khi gặp nghiệm.
  - Nếu không có một khả năng nào chấp nhận được cho x_i hoặc lời giải xấu hơn thì lùi lại bước trước để xác định lại thành phần x_{i-1}.

6. Lược đồ phương pháp

```
Try(i,S)≡ //Sinh thành phần thứ i của cấu hình với chi phí hiện thời S
   for (v thuộc tập khả năng thành phần nghiệm x_i)
     if (v là chấp nhận được)
       T =S + Chi phí nghiệm khi có thêm thành phần v;
       if (T tốt hơn Toptimize) //Tốt hơn chi phí tốt nhất hiện có
          x_i  = v ;
          <Ghi nhận trạng thái chấp nhận v>;
          if(x_i là trạng thái kết thúc)
             <Ghi nhận nghiệm>;
             Toptimize = T; //Cập nhật chi phí tốt nhất
          else
             Try(i+1,T);//Sinh thành phần tiếp theo với chi phí hiện
      thời T
          endif;
             <Khôi phục trạng thái chưa chấp nhận v>;
       endif;
     endif;
   endfor;
End.
```

7. Đánh giá

Việc đánh giá chính xác độ phức tạp của một bài toán có sử dụng phương pháp nhánh cận là không hề đơn giản, tuy nhiên ta cũng có thể ghi nhận một cách hình thức như sau:

- Công thức truy hồi để sinh nghiệm là T(i + 1) = $D_i T(i),\ T(1) = 1$.
- Trong trường hợp xấu nhất ta phải duyệt toàn bộ D = $D_1 \times D_2 \times ... \times D_N$ trạng thái, do đó, trong trường hợp xấu nhất thuật toán có độ phức tạp O($D_1 \times D_2 \times ... \times D_N$).
- Trong trường hợp thuận lợi nhất x = ($v_1^1, v_2^1, ..., v_N^1$) ta cũng cần đến D=N phép kiểm tra điều kiện xem v có chấp nhận được hay không. Độ phức tạp về thời gian trong trường hợp thuận lợi nhất là O(N).

=> Vậy trung bình thuật toán có độ phức tạp là O($D_1 \times D_2 \times ... \times D_N$).

8. Ví dụ

8.1. Bài toán người du lịch

Bài toán người du lịch (Travelling Salesman problem TSP) là một bài toán khá nổi tiếng trong lĩnh vực tối ưu tổ hợp được nghiên cứu trong lý thuyết khoa học máy tính.

8.1.1. Phát biểu bài toán

Có n thành phố (được đánh số từ 1 đến n). Mỗi người du lịch xuất phát từ một thành phố, muốn đi thăm các thánh phố khác, mỗi thành phố đúng 1 lần rồi quay lại nơi xuất phát. Giả thiết giữa 2 thành bất kỳ có thể có hoặc không có đương nối, mỗi đường nối đều có chi phí xác định từ trước. Hãy tìm một hành trình cho người du lịch để có tổng chi phí nhỏ nhất.

8.1.2. Mô tả bài toán

Biểu diễn mạng lưới giao thông giữa các thành phố như một đồ thị có trọng số G = (V, E).

- Mỗi thành phố là một nút của đồ thị (đánh số 1, .. , n).
- Mỗi đường đi giữa các thành phố là một cạnh nối giữa các nút của đồ thị, có thể có hướng hoặc vô hướng, trên đó có ghi trọng số là chi phí đường đi. Các cặp cạnh không có đường đi trọng số là ∞.
- Đỉnh xuất phát ≡ kết thúc: $S \in V$.
- Đường đi tìm được là dãy $S = x_1, x_2,\ldots,x_n, x_1 = S$ với $x_i \in V$, $(x_i,x_{i+1}) \in E$, có tổng chi phí nhỏ nhất.

=> Sinh các dãy hoán vị 1.. *n* và tính dãy có chi phí nhỏ nhất.

8.1.3. Phân tích bài toán

Ở đây chúng ta sẽ đi phân tích bài toán người du lịch giải bằng phương pháp nhánh cận, so sánh với các phương pháp vét cạn, quay lui và tìm cách *cải tiến cận* để tối ưu bài toán.

Bài toán với phương pháp vét cạn: Cách giải trực tiếp nhất có thể là thử tất cả các hoán vị có thể có và xem hoán vị nào là tốt nhất. Có thể thấy ngay được thời gian để chạy là O(n!).

Bài toán với phương pháp quay lui: Cải tiến vét cạn, thêm điều kiện có đường đi giữa 2 đỉnh, ta có thuật toán quay lui. Với cách giải này ta có thể tiết kiệm thời gian không xét đến những trường hợp giữa 2 đỉnh không có đường đi.

Đánh giá phương pháp này: Thời gian để thực hiện = O(số đỉnh có đường đi đến đỉnh 1 × số đỉnh có đường đi đến đỉnh 2 × … × số đỉnh có đường đi đến đỉnh n). Chi phí thực hiện cũng rất lớn và trường hợp xấu nhất bằng với chi phí của thuật toán vét cạn nếu mọi đỉnh đều có đường đi đến các đỉnh còn lại. Như thuật toán quay lui, tại bước thứ i ta chọn một giá trị x_i không tối ưu thì toàn bộ quá trình chọn x_{i+1}, x_{i+2} ... sẽ hoàn toàn vô

nghĩa. Ngược lại, nếu ta xác định được rằng giá trị x_i đó không dẫn đến cấu hình tối ưu thì ta sẽ tiết kiệm được toàn bộ các bước chọn x_{i+1}, x_{i+2} ...***Đây cũng là ý tưởng của thuật toán nhánh cận.*** Với bài toán người du lịch, tại lần di chuyển thứ i, nếu tổng chi phí đang có $\geq$ chi phí của phương án tốt nhất ta đang có thì rõ ràng việc đi tiếp không mang đến kết quả tốt hơn. Do đó, cải tiến tiếp phương pháp quay lui bằng cách đặt một nhánh cận đơn giản như sau:

```
Try(i,C)≡//Sinh thành phần thứ i của cấu hình với chi phí hiện thời C
    for (v = 1..n)
       if ((c[x_{i-1},v]< ∞)&(not Daqua[v])))
            C1 = C + c[x_{i-1},v];
           if(C1 < BestCost)
              x_i = v ;
              Daqua[v] = true;
              if (i = n+1)&(x_i = S)
                 <Ghi nhận nghiệm x_1,x_2…x_{n+1}>;
                 BestCost = C1; //Cập nhật chi phí tốt nhất
              else if (i<=n)
                 Try(i+1,C1); //Sinh thành phần tiếp theo với
chi phí hiện thời C1
              endif;
              Daqua[v] = false;
              endif
           endif;
        endif;
    endfor;
End.
```

<u>Cải tiến cận:</u>

Ta có thể tiếp tục cải thiện cận này bằng cách không chỉ xét chi phí đến thời điểm hiện tại mà còn xét luôn cả chi phí tối thiểu để kết thúc hành trình. Gọi c_{min} là giá trị nhỏ nhất của ma trận C, tương đương với độ dài ngắn nhất của việc di chuyển từ thành phố này đến thành phố kia. Tại bước thứ i ta còn phải thực hiện n – i + 1 bước di chuyển nữa mới kết thúc hành trình (đi qua n – i thành phố còn lại và quay về thành phố 1). Do đó chi phí của cả hành trình sẽ tối thiểu là C1 + (n – i +1)* c_{min}. Nếu chi phí này lớn hơn chi phí của phương án tốt nhất thì rõ ràng phương án hiện tại cũng không thể dẫn đến một phương án tốt hơn. Mã giả có thể được sửa thành:

```
Try(i,C)≡ //Sinh thành phần thứ i của cấu hình với chi phí hiện thời C
  for (v = 1..n)
     if ((c[x_{i-1},v]< ∞)&(not Daqua[v])))
```

```
            C1 = C + c[x_{i-1},v];
            if(C1 + (n-i+1)* c_min < BestCost)
                x_i = v ;
                Daqua[v] = true;
                if (i = n+1)&(x_i = S)
                   <Ghi nhận nghiệm x_1,x_2…x_{n+1}>;
                   BestCost = C1; //Cập nhật chi phí tốt nhất
                else if (i<=n)
                   Try(i+1,C1); //Sinh thành phần tiếp theo với
chi phí hiện thời C1
                endif;
                Daqua[v] = false;
                endif;
            endif;
        endif;
    endfor;
End.
```

8.2. Bài toán dãy ABC

8.2.1. Phát biểu bài toán

Cho trước một số nguyên dương N (N <= 100), hãy tìm một dãy chỉ gồm các kí tự A, B, C thỏa mãn 3 điều kiện:

- Có độ dài N.
- Hai đoạn con bất kỳ liền nhau khác nhau.
- Có ít ký tự C nhất.

8.2.2. Mô tả bài toán

Dựa trên ý tưởng sinh 1 dãy gồm N ký tự được chia thành N giai đoạn. Ở giai đoạn thứ i ta tìm cách đặt 1 trong 3 chữ cái A, B, C vào vị trí thứ i của dãy.

8.2.3. Phân tích bài toán

Ở đây chúng ta sẽ đi phân tích bài toán dãy ABC giải bằng phương pháp nhánh cận, so sánh với các phương pháp vét cạn, quay lui.

Bài toán với phương pháp vét cạn: Tìm tất cả các dãy có độ dài N gồm 3 ký tự A, B, C sau đó kiểm tra xem dãy nào thỏa mãn đề bài: hai đoạn con bất kỳ liền nhau khác nhau, có ít ký tự C nhất.Với cách giải này thời gian thực hiện việc liệt kê tất cả các dãy là là 3^n.

Bài toán với phương pháp quay lui: Cải tiến vét cạn, xây dựng từng thành phần của dãy, tại mỗi bước xây dựng đều kiểm tra xem nếu chấp

nhận đi tiếp theo đường đó thì có thỏa mãn dãy có 2 đoạn con bất kỳ liền kề nhau khác nhau hay không, nếu thỏa mãn thì đi tiếp xây dựng thành phần tiếp theo. Nếu không còn phương án nào để xây dựng thành phần hiện tại thì quay lui, thực hiện lại các bước trước đó.

Sau khi hoàn tất 1 lời giải thì kiểm tra xem lời giải này có tốt hơn lời giải tốt nhất hiện có (dãy ít ký tự C hơn) nếu thỏa mãn thì cập nhật lại lời giải tốt nhất.

Như vậy, so với vét cạn, nếu tại bước xây dựng thành phần thứ i của dãy, nếu điều kiện kiểm tra không thỏa mãn, ta sẽ tiết kiệm 3^{n-i} bước duyệt phía sau.

Bài toán với phương pháp nhánh cận: Cải tiến quay lui, tại mỗi bước chọn thành phần thứ i của dãy, kiểm tra xem nếu đi tiếp theo hướng đó thì có nhận được lời giải tốt hơn lời giải tốt nhất hiện có không, nếu không sẽ không đi theo hướng này, tiết kiệm được thời gian sinh các thành phần tiếp theo theo hướng đó.

<u>Phân tích lời giải:</u>

- Bài toán tìm xâu đưa về tìm nghiệm a = $(a_1, a_2, \dots, a_n)$ trong đó $a_i \in \{A, B, C\}$ và không có hai dãy con liền nhau sao cho số giá trị $a_i = C \to min$.
- Nếu a = $(a_1, a_2, .., a_n)$ là dãy các kí tự thỏa mãn thì rõ ràng trong 4 kí tự liên tiếp phải có ít nhất 1 ký tự C (không chấp nhận 2 ký tự giống nhau đứng cạnh nhau). Do đó với mỗi dãy con có k ký tự liên tiếp của dãy a sẽ luôn có chữ cái C là không nhỏ hơn [k/4].
- Giả sử tại bước chọn ký tự cho a_i ta có C1 ký tự C trong dãy con từ a_1 tới a_i. Khi sinh nốt N – i chữ cái nữa ta cần ít nhất [(N -i)/4] chữ cái C nữa đồng thời gọi BestC là tổng số chữ cái C ít nhất hiện có thì tại bước i việc chọn a_i phải thỏa mãn:
- C1 + [(N – i)/4] < BestC
  Như vậy việc chọn ký tự a_i phải thỏa mãn 2 điều kiện:
  - Dãy con từ a_1 tới a_i không có hai dãy con liên tiếp giống nhau: Sử dụng một hàm Check kiểm tra lần lượt sự giống nhau của hai dãy con liên tiếp có độ dài từ 1 tới [(i + 1)/2] của dãy con từ a_1 tới a_i.
  - C1 + [(N – i)/4] < BestC: Điều kiện đánh giá nhánh cận
- Ban đầu số ký tự C ít nhất có thể là BestC = n.
- Sử dụng giải thuật Try(i) để sinh a_i.
  - Cho a_i nhận lần lượt các giá trị A, B, C.
  - Với mỗi khả năng của a_i tính c1 = số kí tự C khi bổ sung thêm a_i vào xâu a.

- Nếu xâu a thỏa mãn điều kiện không có 2 đoạn con liên tiếp giống nhau và c1 < BestC thì chấp nhận a_i, tiếp tục xác định a_{i+1} bằng cách gọi Try(i+1).
- Nếu i = n ghi nhận lại 1 trạng thái chấp nhận của xâu a, tiếp tục xét các giá trị a_i và cập nhật lại trạng thái tốt nhất của a.

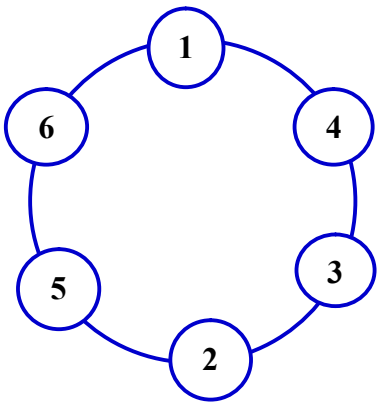
8.2.4. Thuật toán

```
Try(i,S)
{
  for (k = 'A',…,'C')
  {
        a_i = k;
        if(Check(i))
        {
            c_1 = (k ==' C')? S + 1:S;
            if((c_1 + (N - i)/4) < BestC)
            {
                if(i = n)
                {
                    <ghi nhận 1 trạng thái của xâu a>
                    BestC = c_1;
                }
                else
                Try(i+1,c_1);
             }
        }
  }
}
```

8.3. Bài toán vòng tròn số nguyên tố

8.3.1. Phát biểu bài toán

Một vòng tròn chứa 2n vòng tròn nhỏ (Xem hình vẽ). Các vòng tròn nhỏ được đánh số từ 1 đến n theo chiều kim đồng hồ. Cần điền mỗt số tự nhiên từ 1 đến 2n vào một vòng tròn nhỏ (mỗi số chỉ được phép điền một lần) sao cho tổng của hai số trên hai vòng tròn nhỏ liên tiếp là số nguyên tố. Số điền ở vòng tròn nhỏ 1 luôn là 1.



Minh họa: Vòng tròn số nguyên tố

8.3.2. Mô tả bài toán

- Dữ liệu vào cho bởi file VONGSO.INP chỉ gồm duy nhất một dòng ghi số nguyên dương N (1 < N < 10).
- Dữ liệu ra ghi vào file VONGSO.OUT gồm k + 1 dòng.
  - Dòng đầu tiên ghi ra số k là số cách điền số tìm được.
  - k dòng tiếp theo, mỗi dòng ghi ra một cách điền các số vào các vòng tròn nhỏ, bắt đầu từ vòng tròn nhỏ 1, mỗi số cách nhau bởi mốt dấu cách. Cách điền nào có thứ tự từ điển nhỏ hơn thì xếp trước. Nếu k>10000 thì chỉ cần ghi ra 10000 cách đầu tiên.

Ví dụ:

VONGSO.INP	VONGSO.OUT
3	2
	1 4 3 2 5 6
	1 6 5 2 3 4

8.3.3. Phân tích bài toán

Bài toán với phương pháp vét cạn: Liệt kê tất cả các phương án có thể xem phương án nào là nghiệm đúng của bài toán cần giải quyết.

Bài toán với phương pháp quay lui: Bài toán sắp xếp vị trí các số theo hình tròn nên ý tưởng thuật toán xử lý là sẽ sử dụng thuật toán duyệt mọi hoán vị với chỉ số của số hạng đầu tiên là 1. Như vậy việc sinh hoán vị N phần tử coi như việc thiết lập phương án của một dãy công việc gồm có N giai đoạn mà ta đã làm với thuật toán duyệt đệ quy, nhưng có một điểm khác đó là mỗi khi sinh ra một hoán vị ta cần kiểm tra xem nó có thỏa mãn tổng giá trị của hai số liên tiếp có là một số nguyên tố hay không. Nếu mọi vị trí trong dãy đều thỏa mãn thì ta chấp nhận phương án. Vì vậy, sử dụng phương pháp nhánh cận sẽ là phương pháp tối ưu cho bài toán này.

Ý tưởng phương pháp nhánh cận đó như sau:

- Dùng mảng D[i] để đánh dấu xem chỉ số i + 1 đã được sử dụng chưa? Mảng X[i] dùng để lưu phương án, X[0]=1.
- Duyệt đệ quy từ vị trí thứ 2 của mảng X[i] hay gọi đệ quy Try (1).
- Mỗi lần duyệt đến một chỉ số v = 1..N – 1 ta kiểm tra đồng thời v đã được đánh dấu chưa và nếu dùng nó vào vị trí X[i] thì nó cộng với số đứng trước có lập thành số nguyên tố hay không? Nếu không thì bỏ qua.
- Ta cần thêm một hàm kiểm tra số nguyên tố.

8.3.4. Thuật toán

```
void Try(int i)
{
  int v, k;
  for(v = 1; v < N; v++)
  {
     if(D[v]==0 && ngto(X[i-1] + v + 1))//D[v]==0 chua
dc chon
      {
          X[i] = v+1;
          D[v] = 1;    //v đã được chọn
          if (i == N-1)
          {
              if(ngto(X[N-1] + X[0]))
              {
                  for(k = 0; k < N; k++)
                  {
                     <ghi nhận nghiệm>
                     fprintf(fos, "%d ", X[k]);
                  }
                  fprintf(fos, "\n");
              }
          }
          else
              Try (i+1);
              D[v] = 0;
        }
  }
}
```

Hàm kiểm tra nguyên tố:

```
int ngto(int k)
    int i;
    for (i = 2 ; i < k; i++)
    {
       if (k%i == 0)
            return 0; //trả lại giá trị không là số nguyên tố
    }
    return 1; //Trả lại giá trị là số nguyên tố
}
```

9. Kết luận

- Do nằm trong lớp các phương án thử sai nên phương pháp nhánh cận có những nhược điểm chung của lớp.

- So với vét cạn và quay lui thì nhánh cận đã giảm đc thời gian thực hiện tương đối.
- Để giải bài toán tối ưu thì nhánh cận có phải là phương pháp tốt nhất? (cụ thể so với tham lam và quy hoạch động)