**­­­­­­­­­­­­­TRƯỜNG ĐẠI HỌC QUỐC GIA HÀ NỘI**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA : TOÁN – CƠ – TIN HỌC**

**--------------- 🙟🕮🙝 ---------------**



**TIỂU LUẬN GIỮA KÌ**

**Môn học : Thiết kế và đánh giá thuật toán**

**Đề Tài *:***  ***Phương pháp quay lui***

|  |  |
| --- | --- |
| Giáo viên hướng dẫn : | **Nguyễn Thị Hồng Minh** |
| Sinh viên thực hiên : | Lê Thị Ngọc Ánh  Nguyễn Hồng Chương  Lê Thị Hiến  Trần Thị Thu Hằng (4/9) |
| Lớp : | K54A2 Toán Tin |

**Hà Nội : 4 / 2012**

**LỜI NÓI ĐẦU**

Trong quá trình nghiên cứu giải quyết các vấn đề – bài toán, người ta đã đưa ra những nhận xét như sau:

http://cn2.dttx.citd.edu.vn/gtrinh/TH112/Htm/images/home_bullet.gif Có nhiều bài toán cho đến nay vẫn chưa tìm ra một cách giải theo kiểu thuật toán và cũng không biết là có tồn tại thuật toán hay không.

http://cn2.dttx.citd.edu.vn/gtrinh/TH112/Htm/images/home_bullet.gif Có nhiều bài toán đã có thuật toán để giải nhưng không chấp nhận được vì thời gian giải theo thuật toán đó quá lớn hoặc các điều kiện cho thuật toán khó đáp ứng.

http://cn2.dttx.citd.edu.vn/gtrinh/TH112/Htm/images/home_bullet.gif Có những bài toán được giải theo những cách giải vi phạm thuật toán nhưng vẫn chấp nhận được.

Từ những nhận định trên, người ta thấy rằng cần phải có những đổi mới cho khái niệm thuật toán. Người ta đã mở rộng hai tiêu chuẩn của thuật toán: tính xác định và tính đúng đắn. Việc mở rộng tính xác định đối với thuật toán đã được thể hiện qua các giải thuật đệ quy và ngẫu nhiên. Tính đúng của thuật toán bây giờ không còn bắt buộc đối với một số cách giải bài toán, nhất là các cách giải gần đúng. Trong thực tiễn có nhiều trường hợp người ta chấp nhận các cách giải thường cho kết quả tốt (nhưng không phải lúc nào cũng tốt) nhưng ít phức tạp và hiệu quả. Chẳng hạn nếu giải một bài toán bằng thuật toán tối ưu đòi hỏi máy tính thực hiên nhiều năm thì chúng ta có thể sẵn lòng chấp nhận một giải pháp gần tối ưu mà chỉ cần máy tính chạy trong vài ngày hoặc vài giờ.

Các cách giải chấp nhận được nhưng không hoàn toàn đáp ứng đầy đủ các tiêu chuẩn của thuật toán thường được gọi là các thuật giải. Khái niệm mở rộng này của thuật toán đã mở cửa cho chúng ta trong việc tìm kiếm phương pháp để giải quyết các bài toán được đặt ra.

Vét cạn, quay lui ,nhánh cận .. là một số tên gọi tuy không đồng nghĩa nhưng cùng chỉ một phương pháp rất đơn giản trong tin học : *Tìm nghiệm của một bài toán bằng cách xem xét tất cả các phương án có thể.* Đối với con người phương pháp này thường là không khả thi vì số phương án cần kiểm tra quá lớn. Tuy nhiên đối với máy tính, nhờ tốc độ xử lí nhanh, máy tính có thể giải rất nhiều bài toán bằng phương pháp thử sai.

Bài tiểu luận sau đây , sẽ cho chúng ta tìm hiểu rõ hơn về **“ Thuật toán quay lui“** Vì kiến thức của chúng em còn nhiều hạn chế nên trong quá trình làm tiểu luận còn có sai sót. Mong cô và các bạn , góp ý. Để bài tiểu luận của chúng em hoàn chỉnh hơn .

Chúng em xin cảm ơn !

**MỤC LỤC**

|  |  |  |
| --- | --- | --- |
| **I . Tổng quan về phương** | **pháp quay lui** |  |
|  | **1 . Dấu hiệu nhận biết** |  |
|  | **2 . Ý tưởng** |  |
|  | **3 . Mô hình** |  |
|  | **4 . Lược đồ** |  |
|  | **5 . Chứng minh tính đúng** |  |
|  | **6 . Nhận xét** |  |
|  | **7 . So sánh với “ Vét cạn , Nhánh cận“** |  |

|  |  |
| --- | --- |
| **II . Phân tích thiết kế một** | **số bài toán cơ bản** |
|  | 1. **Liệt kê các dãy nhị phân có độ dài n** |
|  | **2. Phân tích số** |
|  | **3. Liệt kê các chỉnh hợp không lặp k phần tử** |
|  | **4. Người bán hàng** |
|  | **5. Bài toán Balo** |
|  | **6. Bài toán xếp Hậu** |
|  | **7. Chu trình Hamilton** |

|  |  |
| --- | --- |
| **III. Một số lỗi mắc phải** | **khi dùng phương pháp quay lui** |
|  | **1 . Lỗi hay mắc phải** |
|  | **2 . Chú ý !** |

|  |  |
| --- | --- |
| **IV. Tổng kết** |  |
|  | **1 . Kết luận** |
|  | **2 . Phụ lục** |

|  |  |  |
| --- | --- | --- |
| **I . Tổng quan về phương** | **pháp quay lui** |  |

**1.Dấu hiệu nhận biết bài toán có thể sử dụng phương pháp**

Một bài toán liệt kê tổ hợp luôn cần phải đảm bảo hai nguyên tắc, đó là: không được bỏ sót một cấu hình và không được trùng lặp một cấu hình. Có thể nói rằng phương pháp liệt kê là cách cuối cùng để có thể giải được một số bài toán tổ hợp hiện nay. Một trong những phương pháp liệt kê có tính phổ dụng cao đó là phươngpháp quay lui*.*

Một số bài toàn liệt kê đơn giản :

* Liệt kê các nước Asean ( Đ ÁN : Việt Nam , Lào , Thài Lan … )
* Liệt kê các số nhị phân 3 bit ( Đ ÁN : 000 , 001 , 010 , 100 , …..)
* Liệt kê các số tự nhiên chia hết cho 5 ( Đ ÁN : 5,10,15,20,25…..)
* Vv …..

**2 . Ý tưởng của phương pháp**

**“** *Quay đầu là bờ “*

Nét đặc trưng của phương pháp lui các bước hướng tới lời giải cuối cùng của bài toán hoàn toàn được làm thử .

Tại mỗi bước , nếu có một lựa chọn được chấp nhận thì ghi nhận lại lựa chọn này và tiến hành các bước thử tiếp theo. Còn ngược lại không có lựa chọn nào thích hợp thì làm lại bước trước, xóa bỏ sự ghi nhận và quay về chu trình thử các lựa chọn còn lại .

Hành động này được gọi là quay lui, thuật toán thể hiện phương pháp này gọi là quay lui.

Điểm quan trọng của thuật toán là phải nghi nhớ mỗi bước đi qua để tránh trùng lặp khi quay lui . Dễ thấy là các thông tin này cần được lưu trữ vào một ngăn xếp , nên thuật toán thể hiện ý thiết kế một cách đệ quy.

**3. Mô hình**

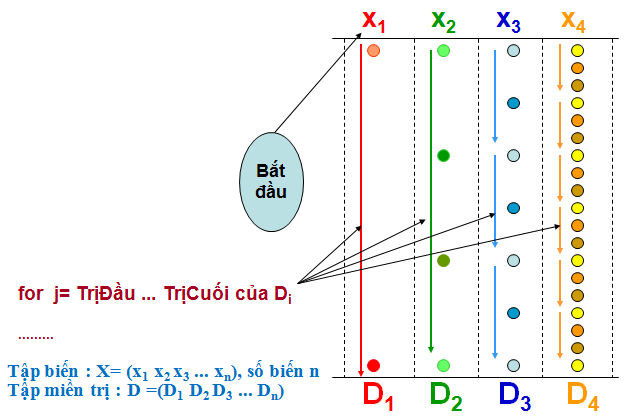
Lời giải của bài toán thường biểu diễn một vec tơ gồm n phần tử = ( ..)

Phải thỏa mãn các điều kiện nào đó. Để chỉ ra lời giải **x**, ta phải xây dựng dần các thành phần lời giải .

Tại bước i :

* Đã xây dựng xong các thành phần  ,…..
* Xây dựng thành phần  bằng cách lần lượt thử cả các khả năng mà  có thể chọn.
  + Nếu một khả năng j nào đó phủ hợp cho  thì ta xác định theo khả năng j .Thường phải có thêm thao tác ghi nhận trạng thái mới của bài toán để hỗ trợ cho bước quay lui. Nếu i = n thì ta có được một lời giải , ngược lại thì tiến hành bước i+1 để xác định .
  + Nếu không có một khả năng nào chấp nhận được cho  thì ta lùi lại bước trước ( bước i -1 ) để xác định lại thành phần 

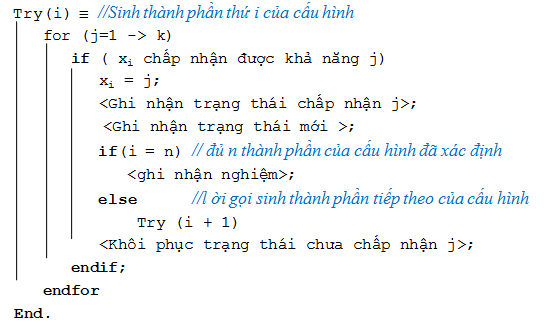
Để đơn giản , ta giả định các khả năng lựa chọn cho các  tại mỗi bước là như nhau , dó đó ta phải có thêm một thao tác kiểm tra khả năng j nào là chấp nhận được cho .



**Hình 1.1 : Mô hình hóa ý tưởng**

**4 . Lược đồ phương pháp**

Mô hình của phương pháp quay lui có thể viết bằng thủ tục sau, với n là số bước cần phải thực hiện , k là số khả năng mà  có thể lựa chọn.



**5 . Chứng minh tính đúng của phương pháp**

Chúng ta sẽ chứng minh tính đúng của phương pháp bằng bất biến vòng lặp

Bất biến vòng lặp : D = {x1 , x2 , … , xn } – Tập các cầu hình.

Ta có 3 điều bất biến về vòng lặp này:

1. Khởi tạo : j=1 có 1 khả năng chấp nhận được nghiệm của 

Bài toán luôn có nghiệm.

1. Duy trì : Tại bước I thuật toán tìm một giá trị cho  , ghi nhận trạng trạng thái rồi gọi đệ quy , đê sinh thành phần . Khi sinh đủ n thành phần của x thì dừng lại cập nhật phương án tối ưu . Nếu mọi khả năng của . đều đã xét qua thì vòng for của try(i+1) thực hiện xong. Sau đó chương trình sẽ quay về đẻ gọi để quy của try (i). Trạng thái cũ trước khi gọi xi được phục hồi và vòng for của try (i) sẽ tiếp tục để chọn giá trị j phù hợp tiếp theo của 
2. Kết thúc : Khi về đến try (1) và xét hết mọi khả năng của x1 thì chương trình con đệ quy sẽ kết thúc.Ta duyệ được tất cả phương án nghiệm. Vậy đúng với mọi vòng lặp và chương trình kết thúc .

**6. Nhận xét**

Tim nghiệm bằng phương pháp quay lui có thể chuyển về tìm kiếm trên cây không gian các trạng thái , với cây được xây dựng từng mức như sau :

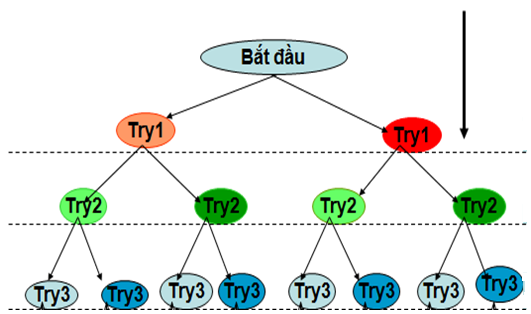
* Các nút con của gốc ( thuộc mức 1 ) là các khả năng có thể chọn cho .
* Gỉa sử  là một nút ở mức thứ i-1 , khi đó các nút con của  là các khả năng mà  có thể chọn , một khi đã tìm được các thành phần  ,…...

Như vậy , mỗi nút  của cây biểu diễn một lời gải bộ phận , đó là các nút nằm trên đường đi từ gốc đến nút đó.

Ta có thể nói việc tìm kiếm nghiệm bằng phương pháp quay lui chính là tìm kiếm theo chiều sâu trên cây không gian các trạng thái.

**Các bài toán có thể giải được bằng phương pháp quay lui:**

* Liệt kê dãy nhị phân độ dài n
* Bài toán quân hậu , mã đi tuần
* Bài toán tìm đường đi trong mê cung
* Tìm chu trình Hamilton của đồ thị (Hamiltonian Path Problem)
* Bài toán người đưa hàng (Traverling Salesman Problem)
* Bài toán xếp balo (Knapsack Problem)
* Bài toán tô màu bản đồ (Map Coloring Problem)
* …..



**Hình 1.2 : Giải thuật tổng quát**

**7 . So sánh Vét cạn – quay lui – nhánh cận**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Vét cạn | Quay lui | Nhánh cận |
| Thời gian | **Rất lớn** | **Ít hơn vét cạn** | **Ít hơn quay lui** |
| Bộ nhớ | **Ít nhất** | **Nhiều hơn vét cạn** | **Nhiều nhất** |
| Độ phức tạp | **Thường là bậc mũ TH xấu nhât là O(n!)** | **Tốt hơn vét cạn( TH xấu nhất vẫn là cấp số mũ)** | **Giảm so với quay lui** |
| Khả năng có nghiệm | **Có** | **Có** | **Có hoặc Không** |

|  |  |
| --- | --- |
| **II . Phân tích thiết kế một** | **số bài toán cơ bản** |

**1.Bài toán Liệt kê các dãy nhị phân có độ dài n.**

**a. Đề bài**

* Liệt kê các dãy nhị phận có dộ dài n .

**b. Phân tích**

* Liệt kê các dãy có chiều dài n dưới dạng  .. trong đó  ∈ { 0,1 } ko cần thỏa mãn điều kiện gì .

**c. Thiết kế thuật toán**

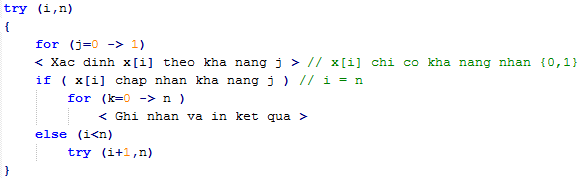
Ta có thể sử dụng sơ đồ tìm tất cả các lời giải của bái toán . Hàm Try(i) xác định  , trong đó  chi có thể nhận một trong hai giá trị 0 hoặc 1.Các gái trị này mặc nhiên được chấp nhận mà không cần phải thỏa mãn điều kiện gì.

Nên hàm try (i) có thể viết như sau :

Trong đó : n là số bước cần phải thực hiện .

k là số khả năng mà  có

Bài toán được khởi tạo với biến i =1

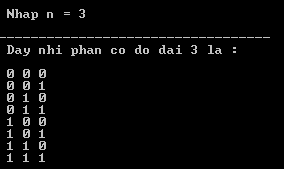
****

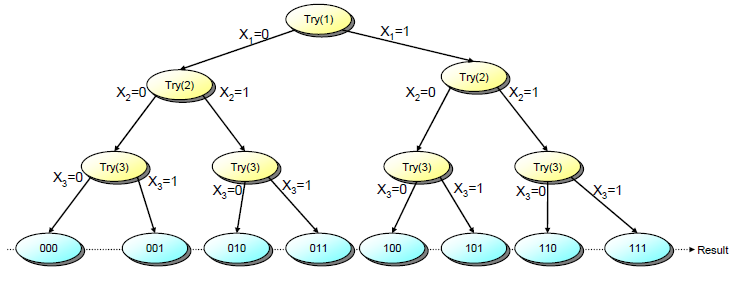
**d. Chương trình**

* *LietKeDayNhiPhan.cpp*
* *LietKeDayNhiPhan.exe*

**e. Ví dụ :**

* **Input :** n = 3
* **Output :**



****

**Hình 2.1 : Mô hình thuật toán in nhị phân ba bit**

**2. Bài toán Phân tích số**

**a . Đề bài**

* Cho số nguyên dương n<=N, hãy tìm tất cả các phân tích số n thành tổng các số nguyên dương , các phân tích là hoán vị của nhau chỉ tính là một.

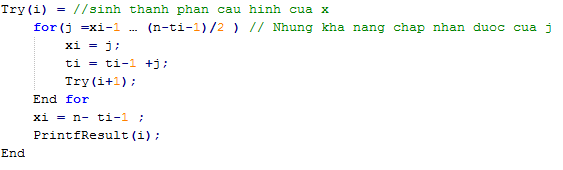
**b. Phân tích**

* Ta sẽ lưu nghiệm trong mảng x, ngoài ra có một mảng t. Mảng t xây dựng như sau: ti là tổng các phần tử trong mảng x từ x1 đến xi .ti = x1 + x2 +…+xi
* Khi liệt kê các dãy x có tổng các phần tử đúng bằng n, để tránh sự trùng lặp ta đưa thêm ràng buôc: xi-1 < =xi.
* Vì số phần tử thực sự của mảng x là không cố định nên thủ tục PrintResult để in ra 1 cách phân tích phải có thêm tham số cho biết sẽ in ra bao nhiêu phần tử.

**c. Thiết kế thuật toán**

* Thủ tục đệ quy Try(i) sẽ thử các giá trị có thể nhận của xi (: xi-1 < =xi).
* Khi nào thì in ra kết quả, khi nào thì gọi đệ quy tiếp ???
* Lưu ý rằng ti-1 là tổng của tất cả các phần tử xi đến xi-1 do đó:
* Khi ti =n tức là (xi = n- ti-1) thì in kết quả.
* Khi tìm tiếp, xi+1 phải thỏa mãn điều kiện : xi+1 >= xi. Mặt khác ti+1 là tổng các các số từ x1 đến xi+1 không vượt quá n. Khi đó, ta có: ti+1 n ⬄ ti-1 + x+xi+1 n ⬄ + xi +xi+1 n- ti-1  xi  ( n- ti-1 n- ti-1)/2.
  + - Ví dụ đơn giản nếu n=10 thì việc chọn x1 = 6,7,8,9 là việc làm vô nghĩa, vì khi đó ta không chọn được x2 được nữa.
* Một cách dễ hiểu ta gọi đệ quy tìm tiếp khi giá trị xi được chọn còn cho phép chọn phần tử khác lớn hơn hoặc bằng nó mà không vượt quá tổng n. Còn ta in kết quả khi xi mang giá trị đúng bằng số thiếu hụt của tổng i-1 phần tử đầu so với n.
* Vây thủ tục Try(i) thử các giá trị cho xi  có thể mô tả như sau (để tổng quát cho i=1, đặt x0 =1,t0 =0).
* Xét các giá trị của xi từ xi-1 đến (n-ti-1)/2, cập nhật ti = ti-1 +xi và gọi đệ quy tìm tiếp.
* Cuối cùng xét giá trị xi = n-ti-1 và in kết quả từ x1 đến xi.
* **Lược đồ :**

Lời gọi ban đầu Try(1)

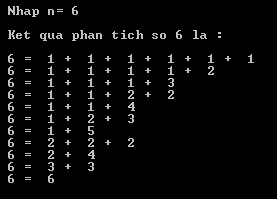
****

**d. Chương trình**

* BaiToanPhantichso.cpp

**e. Ví dụ :**

* **Input :** Nhập **n = 6**
* **Output :**

****

**3.Bài toán liệt kê các chỉnh hợp không lặp chập k**

**a. Đề bài**

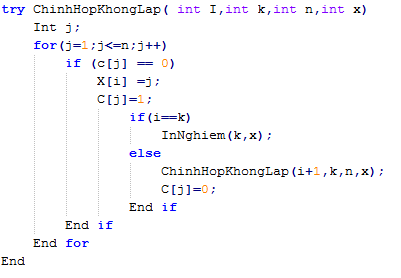
* Cho tập hợp từ 1 đến n .Liệt kê các chỉnh hợp không lặp chập k của n phần tử đã cho.

**b. Phân tích**

* Để liệt kê các chỉnh hợp không lặp chập k của S={1,2,….n } ta có thể đưa về liệt kê các cấu hình (x1,x2,….).Ở đây xi ϵ S và đôi một khác nhau.
* Như vậy thủ tục Try(i)- xét tất cả các khả năng chọn xi-sẽ thử hết các giá trị từ 1 đến n,mà các giá trị này chưa bị phần tử đứng trước chọn.Muốn xem các giá trị nào chưa được chọn ta dung mảng đán h dấu:
* Khởi tạo mảng c1,c2,….,cn mang kiếu logic.ci cho biết giá trị I còn tự do hay đã chọn rồi.Ban đầu khởi tạo các phần tử mảng c la TRUE có nghĩa là các phần tử từ 1 đến n đều tự do.
* Tại bước chọn các giá trị có thể của xi ta chỉ xét nững giá trị j có cj=TRUE nghĩa là chỉ chọn những giá trị tự do.
* Truước khi gọi đệ quy tìm Xi+1 : ta đặt giá trị j vừa gan cho xi là đã bị chọn có nghĩa là đặt cj:= FALSE để các thủ tục Try(i+1),Try(i+2) sau này không chọn phải giá trị j đó nữa.
* Sau khi gọi đệ quy tìm Xi+1 :có nghĩa là sắp tới ta sẽ thử gán một giá trị khác cho xi thì ta sẽ thử đặt giá trị j vừa thử đó thành tự do.(cj=TRUE) bởi khi xi đã nhận giá trị khác rồi thì các phần tử đứng sau có thể nhận gía trị j đó.Điều này hoàn toàn hợp lí trong chỉnh hợp không lặp: x1 có n cách chọn, x2 có n-1 cách chọn…………
* **Lưu ý** rằng khi thủ tục Try(i) có i=k thì ta không cần phải đánh dấu gì cả vì tiếp theo chỉ có in kết quả chứ không cần phải chọn them phần tử nào nữa.

**c.Thiết kế thuật toán**

* Khác với chỉnh hợp lặp là các thành phần được phép lặp lại, tức là có thể giống nhau, chỉnh hợp không lặp chập k của tập n phần tử cũng là một dãy k thành phần lấy từ tập n phần tử có xét thứ tự nhưng các thành phần không được phép giống nhau.
* Nghiệm của bài toán **tìm các chỉnh hợp không lặp chập k của tập n số nguyên từ 1 đến n** là các vector x thoả mãn các điều kiện:
* x có k thành phần: x = (x1,x2,…xk)
* Các giá trị xi lấy trong tập {1,2,..n}
* Ràng buộc: các giá trị xi đôi một khác nhau, tức là xi≠xj với mọi i≠j.
* Ta sẽ sử dụng mảng c[ ] dùng để dánh dấu.
* Dùng thủ tục try(i) để tìm.Giả sử ta đang ở vị trí thứ i nào đó và độ dài của chuỗi số <=k.Nếu i=k thì in kết quả ra màn hình,còn nếu i<k thì lại tiếp tục tìm tiếp.
* Quá trình lặp lại cho đến khi in hết nghiệm.
* **Chú ý:** Dùng 1 biến j chạy từ 1->n, nếu j chưa được chọn thì thực hiện chấp nhận j và đánh dấu lại j đã được chọn.Sau mỗi bước gọi lại thủ tục , đánh dấu lại j chưa được chọn.
* **Lược đồ :**

****

**d.Chương trình**

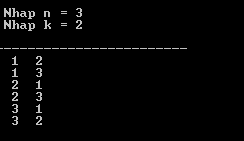
***ChinhHopKhongLapChapk.cpp***

**e.Ví dụ**

* **Input :** Nhap n = 3

k= 2

* **Output :**



**4.Bài toán “ Người bán hàng “**

**a. Đề bài**

* Một người bán hàng trên hệ thống *n* thành phố.
* Giữa các thành phố có thể có hoặc không các đường nối, mỗi đường nối có chi phí xác định từ trước.
* Người bán hàng xuất phát từ một thành phố, đi tới tất cả các thành phố khác và mỗi thành phố đi qua một lần và quay trở lại thành phố ban đầu.
* Hãy xác định một hành trình sao cho tổng chi phí trên đường đi là nhỏ nhất.

**b. Phân tích**

* + 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* ∈*V*
    - Đường đi tìm được là dãy *S*=*x*1,*x*2,…,*x*n,x1=*S* với *x*i ∈*V*, (*x*i,*x*i+1) ∈*E,* có tổng chi phí nhỏ nhất
    - Sinh các dãy hoán vị 1..*n* và tính dãy có chi phí nhỏ nhất:
  + **Tiếp cận theo phương pháp quay lui**
* **Vét cạn:**
* Sinh hoán vị n đỉnh.
* Mỗi hoán vị tính tổng chiều dài
* Tìm hoán vị có tổng min
* **Quay lại:**
* Dùng try(i) để sinh đỉnh
* Nếu *i*=*n* thì in giá trị nghiệm, ngược lại sinh tiếp *x*i+1 bằng Try(*i*+1)
* Tìm được 1 nghiệm, tính tổng chiều dài
* So sánh với tổng chiều dài ngắn nhất hiện có
* Cập nhật lại nếu cần

**c. Thiết kế thuật toán**

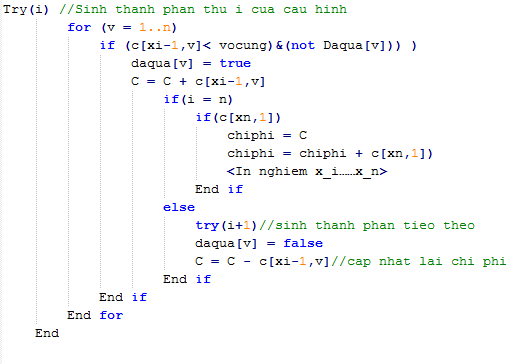
* + **Mô tả dữ liệu:**
* Đánh chỉ số các đỉnh của đồ thị từ 1..*n*
* Biểu diễn đồ thị G bằng ma trận kề *M* = (*c*ij) cỡ *n* x *n*
* *c*ij =  const nếu có cạnh nối đỉnh *i* với đỉnh *j*

nếu ngược lại

* Mảng: Daqua[1..*n*] đánh dấu đỉnh *i* đã được đi qua trên đường đi hay chưa?
* Daqua[*i*] = *true* nếu đỉnh *i* đã có trên đường đi

*false* nếu ngược lại, đỉnh *i* chưa có trên đường đi

* Khởi tạo: Daqua[1..*n*] = *false*
  + **Lược đồ:**

****

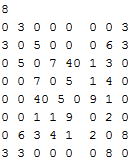
**d. Chương trình**

* + File input: *NguoiBanHang.txt*
  + File chương trình:  *NguoiBanHang.cpp*

e. Ví dụ

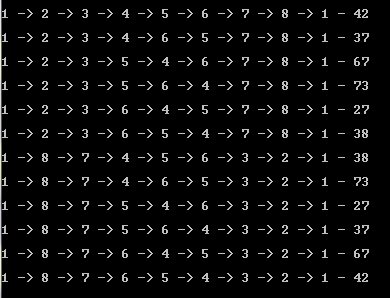
* Đồ thị có trọng số biểu diễn đường đi giữa các thành phố.Với n đỉnh
  + Input : *NguoiBanHang.txt*

Ma trận biểu diễn đồ thị: matran[max][max]



* + **Output:**

**Các đường đi - Chí phí**

****

**5. Bài toán “ Ba lô”**

**a. Đề bài**

* + Cho N đồ vật
  + Mỗi đồ vật có trọng lượng P và giá trị V
  + Ba lô có giới hạn trọng lượng W
  + Tìm cách xếp các vật vào ba lô sao cho tổng trọng lượng không vượt quá giới hạn trọng lượng của ba lô và
  + tổng giá trị là lớn nhất.

**b. Phân tích**

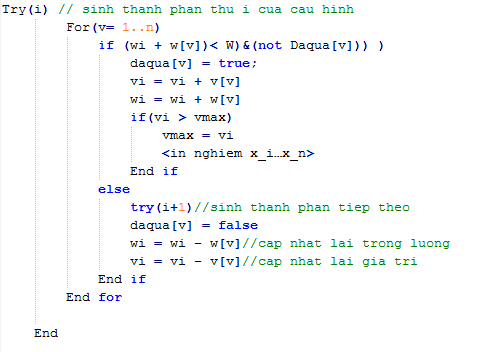
* Trọng lượng tối đa của ba lô là W
* Dùng try(i) để sinh
* Tại mỗi bước chọn nếu chưa cho vật vào ba lô và ba lô chưa đạt trọng lượng tối đa
* Thì cho vật vào ba lô và cập nhật giá trị và trọng lượng mới
* Nếu giá trị mới > giá trị max thì giá trị max = giá trị mới
* Nếu *i*=*n* thì in giá trị nghiệm ngược lại sinh phần tử bằng try(i+1)
* Cập nhật lại giá trị và trọng lượng của ba lô
* Kết thúc khi ba lô đạt trọng lượng tối đa

**c. Thiết kế thuật toán**

* **Mô tả dữ liệu:**
* Trọng lượng tối đa là W,giá trị mới vi, trọng lượng mới wi
* Mảng: Daqua[1..*n*] đánh dấu vật *i* đã được cho vào ba lô hay chưa?
* Daqua[*i*] = *true* nếu đỉnh *i* đã có trong ba lô

*false* nếu ngược lại, đỉnh *i* chưa có trong ba lô

* Khởi tạo: Daqua[1..*n*] = *fals*
* **Lược đồ:**

**

**d. Chương trình**

* **File input:** *balo.txt*
* File chương trình: *balo\_quaylui.cpp*

e. Ví dụ

* **Input ( file :** *balo.txt* )

:

|  |  |
| --- | --- |
| C:\Users\Lioness\Desktop\8.PNG |  |

* **Output**
* Phương án chọn 3 vật :
* **1, 2,4** có tổng trọng lượng **= 4** không vượt quá giới hạn trọng lượng của ba lô và tổng giá trị đạt tối đa là **6**

**6. Bài toán Xếp Hậu**

1. **Đề bàì**

* Xét bàn cờ tổng quát kích thước n\*n. Một quân hậu trên bàn cờ có thể ăn được các quân khác tại các ô cùng hàng, cùng cột hoặc cùngđường chéo.
* Hãy tìm cách xếp n quân hậu trên bàn cờ sao cho không quân hậu nào ăn quân hậu nào.

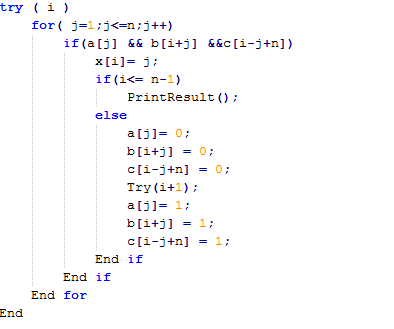
1. **Phân tích**

* n quân hậu sẽ được đặt mỗi con một hàng vì hậu ăn được hàng ngang, gọi quân hậu sẽ đặt hàng i là quân hậu i ( i chạy từ 1 đến n).
* Một nghiệm của bài toánđược biết khi ta tìm ra được vị trí cột của những quân hậu
* Ta định hướng Đông (Phải),T ây (trái), Nam (dưới ), Bắc (trên):
* Một đường chéo theo hướng Đông Bắc –Tây Nam (ĐB–TN) bất kỳ sẽ qua 1 số ô, các ô có tính chất: hàng + cột = C1(const) . Với mỗi đường chéo xác định 1 hằng số duy nhất : 2 ≤ C1 ≤2n. Do đó ta có thể đánh chỉ số cho các đường chéo ĐB–TN từ 2 đến 2n.
* Một đường chéo hướng Đông Nam –Tây Bắc (ĐN-TB) bất kỳ sẽ qua một số ô, các ô có tính chất: hàng–cột = C2. Tương tự mỗi một đường chéo (ĐN –TB) xác định duy nhát một chỉ số C2: 1-n≤C2≤n–1. Ta đánh chỉ số cho các đường chéo ĐN–TB từ 1 đến 2(n-1) +1.

1. **Thiết kế thuật toán**

* Ta dùng 3 mảng lôgic để đánh dấu:
* Mảng agồm n phần tử , ai= TRUE nếu cột i còn tự do, ai= FALSE nếu cột i đã có 1 quân hậu
* Mảng bgồm 2n-1 phần tử: bi = TRUE nếu đường chéo ĐB-TN thứ i còn tự do, ngược lại bi= FALSE .
* Mảng cgồm 2(n-1) +1 phần tử: ci= TRUE nếu đường chéo ĐN - TB thứ i còn tự do , ngược lại ci= FALSE
* Banđầu 3 mảng được khởi tạo mang giá trị TRUE
* Thuật toán quay lui:
* Xét tất cả các cột, thử đặt quân hậu 1 vào một cột, với mỗi cách đặt nhưvậy , xét tất cả các cách đặt quân hậu 2 không bị quân hậu 1 ăn, thử 1 cách đặt quân hậu 2 và sauđó lại xét tiếp các cách đặt quân hậu 3. Mỗi cách đặt được đến quân hậu n cho ta 1 nghiệm
* Khi chọn vị trí cột j cho quân hậu i thì ta phải chọn ô (i,j) không bị các quân hậu đặt trước đó ăn. Tức chọn cột j còn tự do, đường chéo ĐB –TN ( i + j ) tự do, đường chéo ĐN –TB ( i–j + n) tự do. Tức ta kiểm tra điều kiện aj= bi+j= cij+n=TRUE
* Khiđặt được quân hậu i vào cột j, nếu i = n thì in nghiệm. Nếu không:
* Trước khi gọi đệ quy tìm cách đặt quân hậu i + 1 ta đánh dấu cột và 2đường chéo bị quân hâu i khống chế ( aj=bi+j= ci-j+n= FALSE )để lần gọi đệ quy tiếp sau chọn các quân hậu kế tiếp không chọn vào những ô nằm trên cột j và những đường chéo đó.
* Sau khi gọi đệ quy tìm cách đặt quân hậu i + 1, tức là sắp tới ta thử một cách đặt khác cho quân hậu i , ta bỏ đánh dấu cột và 2đường chéo vừa bị thử đăt khống chế ( ai= bj+j=cj-j+n= TRUE ) tức cột và 2 đường chéo đó tự do vì khi đặt quân hậu i sang vị trí khác thì cột và 2 đường chéo đó có thể đặt một quân hậu khác.

**Lược đồ :**

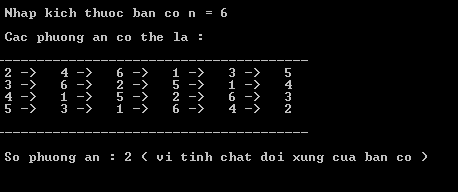
****

**d.Chương trình**

* *BaiToanXepHau.cpp*

**e.Ví dụ**

* **Input** : Nhập kích thước bàn cờ **n = 6**
* **Output :**

****

**7. Bài toán Chu trình Hamilton**

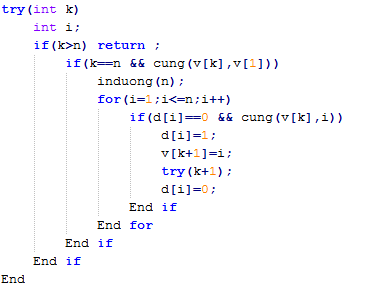
**a. Đề bài** .

* Tìm chu trình Hamilton.
* *Chu trình Hamilton:* Là chu trình đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần và một cạnh trong đồ thị nối đỉnh đầu của dây chuyền với đỉnh cuối của nó.

**b. Ý tượng** :

* Đưa đỉnh x vào đường đi.
* Kiểm tra chu trình đủ chiều dài chưa? (tức là đã đi hết tất cả các đỉnh chưa?) và đồng thời kiểm tra đình đầu và đỉnh cuối của chu trình có cạnh nối hay ko? Nếu thoả mãn hết cả 2 điều kiện thì kết luận có chu trình Hamilton.
* Duyệt hết tất cả các đỉnh kề với x.
* Kiểm tra đỉnh kề đã đi qua chưa? Nếu true gọi lại hàm đệ qui tìm chu trình Hamilton.
* Xóa đỉnh kề ra khỏi chu trình

**c. Thiết kế thuật toán**

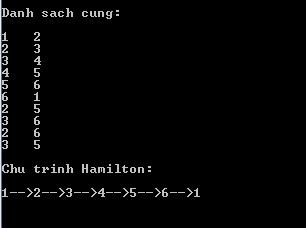
****

**d. Chương trình**

* **File input** *: dt.dat*
* **File Chương Trình** *: ChutrinhHamilton.cpp*

**e.Ví dụ**

* **Input *:*** *dt.dat*
* **Output:**

****

|  |  |
| --- | --- |
| **III. Một số lỗi mắc phải** | **khi dùng phương pháp quay lui** |

**1 . Lỗi mắc phải**

* **Lỗi trong bài toán**
* Khi xác định bài toán dễ nhầm với các bài toán khác trong cùng phương pháp thử sai
* **Lỗi trong lặp trình**
* Khi lập trình có thể sẽ tạo vòng lặp vô hạn
* Không biết lần gọi đệ quy tương ứng với nút nào trên cây.

**2 .Chú ý !**

* Không phải cấu hình nào cũng được sinh ra từ cấu hình trước đó một cách dễ dàng.
* Phương pháp sinh kế tiếp chỉ giải quyết được các bài toán đơn giản.
* Không phải cấu hình ban đầu và cấu hình kế tiếp được nhận diện một cách dễ dàng,nhiều khi phải chứng minh tồn tại chúng.

|  |  |
| --- | --- |
| **IV. Tổng kết** |  |

**1.Kết luận**

* **Ư u điểm :**
* Việc quay lui là thử tất cả các tổ hợp để tìm được một lời giải .Thế mạnh của phương pháp này là nhiều cài đặt tránh được viêc phải thử nhiều trường hợp chưa hoàn chỉnh, và nhờ đó giảm thời gian chạy.
* **Nhược điểm :** 
  + Trong trường hợp xấu nhất độ phức tạp của quay lui. Vẫn là cấp số mũ .Vì nó mắc phải những nhược điểm sau.
* Rơi vào tình trạng “ thrashing” : quá trình tìm kiếm cứ gặp phải bế tắc với cùng một nguyên nhân .
* Thực hiện các công việc dư thừa : Mỗi lân chúng ta quay lui.Chúng ta phải đánh giá lại lời giải trong khi đôi lúc điều này không phải cần thiết .
* Không sớm phát hiện được khả năng bị bế tắc trong tương lai. Quay lui chuẩn , không có cơ chế nhìn về tương lai.Để nhận biết được nhánh đang tìm kiếm sẽ đi vào bế tắc .

**2.Phụ lục**

* **Tài liệu tham khảo.**
* Sile bài giảng thiết kế và đánh giá thuật toán của cô **Nguyễn Thị Hồng Minh**
* Giải thuật và lập trinh . **Lê Minh Hoàng**
* Bài giảng thiết kế và đánh giá thuật toán . Khoa CNTT trường ĐH Giao thông vận tải Hà Nội
* Và kiến thức từ nguồn Internet
* **Phân công công việc**

|  |  |  |
| --- | --- | --- |
| Họ và tên | Lý Thuyết | Bài Tập |
| **Nguyễn Hồng Chương** | **Dấu hiệu và ý tưởng** | **In dãy nhị phân độ dài n** |
| **Lê Thị Ngọc Ánh** | **Mô hình và lược đồ** | **Phân tích số và tổ hợp chập k** |
| **Lê Thị Hiến** | **Chứng minh và nhận xét** | **Balo và người bán Hàng** |
| **Trần Thị Thu Hằng (4/9)** | **So sánh** | **Bài toán xếp hậu và Hamilton** |
| **Cả Nhóm** | **Làm phần III và IV** | **Tìm tài liệu và hỗ trợ cho các thành viên khác trong nhóm** |