Bài 2: Phân tích bài toán, xây dựng thuật toán và viết chương trình

Bài 2.1 Liệt kê các hoán vị của n phần tử

Bài làm:

1. Phân tích bài toán

Ta cần liệt kê tất cả các hoán vị có chứa đủ n phần tử đề bài cho. Trong đó, mỗi phần tử chỉ được xuất hiện một lần duy nhất trong mỗi hoán vị. Mỗi lần ta sắp một phần tử vào hoán vị, nếu phần tử nào đã được sắp vào hoán vị rồi, thì sẽ không được sắp vào hoán vị ấy nữa.

1. Thuật toán

* Biểu diễn các hoán vị dưới dạng p1, p2, …, pn trong đó mỗi pj nhận giá trị từ 1 đến n và pi ≠pj
* Các phần tử từ 1 đến n được đề cử cho pj , j sẽ được chấp nhận nếu giá trị j chưa từng được dùng trước đó. Dùng một mảng để đánh dấu những phần tử đã được dùng b[j]=1 nếu j chưa được dùng và ngược lại. Ban đầu, b[j]=1. Sau khi gán j cho pj, b[j]=0. Khi gọi lại hàm hoanvi(i+1), b[j]=1 để thực hiện phép thủ tiếp theo.

1. Chương trình : ***hoanvi.c***

2.2 Liệt kê các tổ hợp k phần tử của các số từ 1 đến n.

- Phân tích:

x là tổ hợp k phần tử của các số từ 1 đến n.

+ Không gian nghiệm: D{(x)} x = (x1,x2,…,xk)

+ Quy tắc xác định nghiệm: x = (x1,x2,…,xk) thỏa mãn

1. 1 <= xi <= n, i = 1…k
2. xi < xi+1 , i=1…k

(Vì tổ hợp đó không phân biệt thứ tự các phần tử nên ta sắp xếp các phần tử theo thứ tự tăng dần)

* Lược đồ giải thuật:

Try(i)≡

for (j = x[i-1]+1…n )

x[i] = j;

if( i=k ) printResult(x);

else Try(i+1)

endfor

End.

* 1. Bài toán xếp quân cờ:

1. Hãy chỉ ra một cách xếp 8 con hậu trên bàn cờ vua sao cho không con nào khống chế được con nào ( con hậu có thể khống chế các con cờ khác trên cùng hàng ngang, hàng dọc hoặc đường chéo).

* Phân tích:

x là một phương án sắp xếp các con hậu trên bàn cờ.

x = (x1,x2,…,x8) \_ xi: vị trí sắp xếp con hậu thứ i ở cột thứ i.

+ Không gian nghiệm: D{(x)}

+ Quy tắc xác định nghiệm:

1. xi khác nhau từng đôi một.

2. Không có hai con hậu nào ở trên cùng một đường chéo.

- Mô tả dữ liệu: ta dùng 4 mảng

+ mảng x[]: x[i] là cột ta đặt con hậu thứ i.

+ mảng cot[]: cot[j]=1 nếu cột j đã có một con hậu được đặt, ngược lại thì cot[j]=0.

+ mảng dcc[]: dcc[k]=1 nếu đường chéo chính thứ k đã có một con hậu được đặt, tức là ta đã đặt một con hậu tại vị trí (i,j) mà i−j=k; ngược lại thì dcc[k]=0.

+ mảng dcp[]: dcp[k]=1 nếu đường chéo phụ thứ k đã có một con hậu được đặt, tức là ta đã đặt một con hậu tại vị trí (i,j) mà i+j=k, ngược lại thì dcp[k]=0.

* Lược đồ giải thuật:

Try( i )≡

for(j= 1…8)

if((cot[j]=0)&&(dcc[i-j]=0)&&(dcp[i+j]=0))

x[i] = j;

cot[j] = 1;

dcc[i-j] = 1;

dcp[i+j] = 1;

if(i=8)

printf("%d\n",dem++);

printResult(x);

else Try(i+1);

endif

cot[j] = 0;

dcc[i-j] = 0;

dcp[i+j] = 0;

endif

endfor

End.

1. Hãy chỉ ra một cách đi của con mã để xuất phát từ một ô cờ (x,y) nào đó (x,y=1…8) có thể đi qua tất cả các ô cờ, mỗi ô duy nhất một lần.
   1. Bài toán xâu ABC: Cho số nguyên dương n<100, tìm một xâu gồm toàn các kí tự A, B, C thỏa mãn: Xâu có độ dài n, 2 đoạn con bất kì liền nhau đều khác nhau, xâu có ít kí tự C nhất.

- Phân tích :

x là xâu thỏa mãn yêu cầu đề bài.

x = (x1,x2,…,xn)

+ Không gian nghiệm: D{(x)}

+ Quy tắc nghiệm:

1. xi là các kí tự A, B, C.
2. Không có 2 đoạn con bất kì liền nhau giống nhau.
3. Xâu có ít kí tự C nhất.

* Ý tưởng thuật toán:

+ Tại bước thứ i ta sẽ tìm cách đặt một trong ba chữ cái A, B, C vào vị trí thứ i của dãy.

+ Vì không có 2 đoạn con liên tiếp nào giống nhau nên trong 4 kí tự liên tiếp phải có một kí tự C. Vậy trong k kí tự liên tiếp số kí tự C luôn lớn hơn hoặc bằng k/4.

+ Gọi t là số kí tự C tại thời điểm thứ i, để sinh nốt xâu còn lại ta cần ít nhất (n-i)/4 kí tự C nữa. Do đó ta có điều kiện: t+(n-i)/4 < cmin. ( Trong đó: cmin là số kí tự C ít nhất sau khi chuỗi được sinh ra. Ban đầu ta gán: cmin = n)

* Mô tả dữ liệu:

+ mảng x[]: lưu trữ xâu gồm các kí tự A, B, C.

+ mảng xBest[]: lưu trữ xâu gồm các kí tự A, B, C tốt nhất thỏa mãn yêu cầu.

+ cmin: số kí tự C có trong dãy.

* Lược đồ giải thuật:

Try( i, s)≡ // i: kí tự thứ i; s: số kí tự C tại thời điểm i.

for(v=A…C)

x[i] = v;

if(kiem\_tra(i))

t = (v=='C')?s+1:s;

if((t+(n-i)/4)<cmin)

if(i=(n-1))

cmin = t;

for(j=0;j<n;j++)

xBest[j] = x[j];

else Try(i+1,t);

endif

endif

endif

endfor

End.

Bài 3: Tự đặt một số đề bài toán, phân tích bài toán, xây dựng giải thuật, đánh giá độ phức tạp và viết chương trình để minh họa phương pháp quay lui, phương pháp nhánh cận (mỗi phương pháp ít nhất 01 bài toán).

## Bài toán 1: ( Phương pháp quay lui)

Liệt kê tất cả đường đi từ đỉnh i đến đỉnh j trong đồ thị có hướng.

1. Phân tích bài toán:

Ta cần liệt kê tất cả các đường đi từ đỉnh i đến đỉnh j trong một đồ thị có hướng. Trong đồ thị, có thể có nhiều đường đi từ đỉnh i đến đỉnh j, cũng có thể có một đường đi duy nhất hoặc không có đường đi nào. Nếu tồn tại đường đi giữa hai đỉnh i và j thì đường đi này chỉ đi qua một đỉnh một lần duy nhất.

1. Thuật toán.

* Đường đi từ i đến j là tập các đỉnh ai , ai nhận giá trị từ 1 đến n, và mỗi ai đều khác nhau.
* Các đỉnh từ 1 đến n được đề cử cho ai , i sẽ được chấp nhận nếu đỉnh i chưa từng được đi qua trước đó và tồn tại cung (ai-1, ai). Dùng một mảng để đánh dấu những phần tử đã được dùng b[i]=1 nếu i chưa được đi qua và ngược lại. Ban đầu, b[j]=1. Sau khi gán i cho ai, b[j]=0. Khi gọi lại hàm lietke(i+1), b[i]=1 để tìm đường đi tiếp theo nếu có.

1. Chương trình: ***lietke.c***
2. Đánh giá độ phức tạp:

## Bài toán 2: ( Phương pháp nhánh cận)

Bài toán ba lô: Một kẻ trộm đột nhập vào một cửa hiệu tìm thấy có *n* mặt hàng có trọng lượng và giá trị khác nhau, nhưng hắn chỉ mang theo một cái túi có sức chứa về trọng lượng tối đa là *M*. Vậy kẻ trộm nên bỏ vào ba lô những món nào và số lượng bao nhiêu để đạt giá trị cao nhất trong khả năng mà hắn có thể mang đi được.(giả sử mỗi loại mặt hàng chỉ có 1 sản phẩm).

1. Phân tích bài toán:

* Ta có n loại đồ vật x1, x2, …, xn trong đó mỗi loại có trọng lượng pj, j=1,2,…n.
* Yêu cầu là cần sắp xếp, lựa chọn sao cho giá trị của đồ vật có thể đem đi là lớn nhất mà vẫn đảm bảo khối lượng tối đa không bị vượt quá.

1. Thuật toán:

* Ta có thể coi như ta đang xét xem vật nào có tỉ lệ giữa giá trị và trọng lượng cao nhất sẽ được chọn.
* Dùng một mảng để đánh dấu những vật đã được xét, vật k sẽ được tiếp nhận nó có giá trị cao và khối lượng không quá lớn, có nghĩa là tỉ số giữa giá trị và khối lượng cao. Sau khi đã xét xong, cần trả lại vật k chưa được đánh dấu. Nếu không thể thêm thì quay lại và xét lại vật đã thêm trước đó. Nếu đã thêm rồi, ta cũng kiểm tra lại phần tử thêm trước xem có tốii ưu không.

1. Chương trình: ***balo.c***
2. Đánh giá độ phức tạp: