2.1.2 Thuật toán quay lui (backtracking)

Created by TUNG DUC NGUYEN tung2.nguyen, last modified on 2021/05/12

1. Thuật toán quay lui

Thuật toán quay lui dùng để giải bài toán liệt kê các cấu hình. Mỗi cấu hình được xây dựng bằng cách xây dựng từng phần tử, mỗi phần tử được chọn bằng cách thử tất cả các khả năng.

Giả sử cấu hình cần liệt kê có dạng (x1, x2, ..., x(n)). Khi đó thuật toán quay lui thực hiện qua các bước sau:

1> Xét tất cả cá giá trị x1 có thể nhận, thử cho x1 lần lượt nhận các giá trị đó. Với mỗi giá trị thử gán cho x1 ta sẽ tiến hành bước 2.

2> Xét tất cả cá giá trị x2 có thể nhận, thử cho x2 lần lượt nhận các giá trị đó. Với mỗi giá trị thử gán cho x2 ta sẽ tiến hành các khả năng chọn cho x3... cứ tiếp tục như vậy cho đến bước n

.....

n> Xét tất cả cá giá trị x(n) có thể nhận, thử cho x(n) lần lượt nhận các giá trị đó. Với mỗi giá trị thử gán cho x(n) ta sẽ thông báo cấu hình tìm được (x1, x2, ..., x(n)).

Trên phương diện quy nạp, ta có thể nói rằng thuật toán quay lui liệt kê các cấu hình n phần tử dạng (x1, x2, ..., x(n)) bằng cách thử cho x1 nhận lần lượt các giá trị có thể. Với mỗi giá trị gán cho x1 lại liệt kê cấu hình n-1 phần tử (x2, x3, ..., x)n)).

Mô hình của thuật toán quay lui có thể mô tả như sau:

```
void try(int i) {
    for (v: {tập giá trị V mà có thể gán cho x[i]}) {
        x[i] = v;
        if (i == n) {
            thông báo cấu hình tìm được;
        }
        else {
            mark[v] = true; //Ghi nhận việc x[i] nhận giá trị v nếu cần
            try(i + 1);
            mark[v] = false; //Bỏ việc ghi nhận x[i] nhận giá trị v nếu cần
        }
    }
}
```

Thuật toán quay lui sẽ bắt đầu bằng lời gọi try(1);

2. Liệt kê các dãy nhị phân độ dài n (bài toán 4)

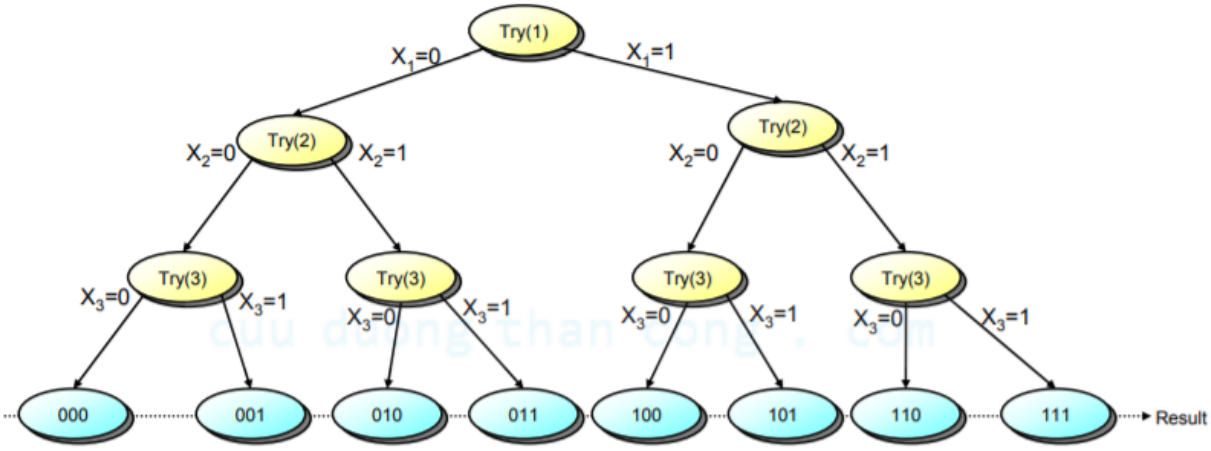
Dãy nhị phân đội dài n được biểu diễn dưới dạng (x1, x2, ..., x(n)). Ta sẽ liệt kê dãy này bằng cách dùng các giá trị {0 , 1} gán cho x(i). Với mỗi giá trị thử gán cho x(i) ta thử lại các giá trị có thể gán cho x(i+1).

Ta có chương trình liệt kê các dãy nhị phân độ dài n bằng thuật toán quay lui như sau:

```
void try(int i) {
```

```
    for (int v = 0; v <= 1; ++v) {
        x[i] = j;
        if (i == n)
            print(x);
        else
            try(i + 1);
    }
}
```

Khi n =3, cây tìm kiếm quay lui sẽ như sau:



⭐ Với dãy tam phân, tứ phân, ... n phân thì việc áp dụng cũng chỉ đơn giản là thay tập giá trị V. Với nhị phân V = {0 , 1}, tam phân V = {0 , 1, 2}, ... n phân V = {0 , 1, ..., n}

3. Liệt kê các tập con k phần tử (bài toán 2)

Để liệt kê tập con k phần tử của tập {1, 2, ..., n}, ta có thể đưa về liệt kê các cấu hình (x1, x2, ..., x(k)) với x1 < x2 < .... < x(k).

Ta có nhận xét:

x(k) ≤ n

x(k-1) ≤x(k) - 1 ≤ n -1

....

x(i) ≤ n - k + i

....

x1 ≤ n -k + 1

Từ đó suy ra x(i-1) + 1 ≤ x(i) ≤ n - k + i (1 ≤ i ≤ k), ở đây giả thiết chúng ta có thêm 1 số x0 khi xét i = 1.

Như vậy ta sẽ xét tất cả các cách chọn x1 từ 1 (x0 + 1) đến n - k + 1, với mỗi giá trị đó xét tất cả các lựa chọn x2 từ x1+1 đến n - k + 2, ..., cứ như vậy đến x(k) thì ta có cấu hình cần liệt kê.

Ta có chương trình liệt kê các tập con k phần tử bằng thuật toán quay lui như sau:

```
void try(int i) {
    for (int v = x[i - 1] + 1; v <= n - k + i; ++v) {
        x[i] = v;
        if (i == k)
```

```
            print(x);
        else
            try(i+1);
    }
}
```

4. Liệt kê các chỉnh hợp không lặp chập k (bài toán số 3)

Để liệt kê các chính hợp không lặp chập k của tập {1, 2, ..., n} ta có thể đưa về liệt kê các cấu hình (x1, x2, ..., x(k)) với các x(i) khác nhau từng đôi một.

Như vậy hàm try(i) xét tất cả các khả năng chọn từ 1 đến n mà các giá trị này chưa bị các phần tử đứng trước chọn. Muốn xem phần tử nào chưa được chọn thì chúng ta sử dụng mảng đánh dấu.

Khởi tạo một mảng c1, c2, ..., c(n) kiểu boolean. C(i) cho biết giá trị i có còn tự do hay đã bị chọn rồi.

Ban đầu, tất cả giá trị trong mảng c là TRUE, có nghĩa là đang tự do.

Tại bước chọn các giá trị có thể của x(i), ta chỉ xét những giá trị đang tự do.

Trước khi gọi đệ quy tìm x(i+1) <try(i + 1)>, ta cần đặt giá trị v vừa gán cho x(i) là đã bị chọn, có nghĩa là đặt c(v) = FALSE để các bước tiếp theo try(i+1), try(i+1),... không chọn phải giá trị đó nữa.

Sau khi gọi đệ quy tìm x(i+1) <try(i + 1)>, nghĩa là chúng ta **chuẩn bị gán một giá trị khác** cho x(i) thì chúng ta đặt lại giá trị v vừa thử thành **tự do**, có nghĩa là c(v) = TRUE.

Tại bước i = k, ta không cần đánh dấu vì sau đó không còn bước nào nữa cả, việc của chúng ta là in ra cấu hình.

Ta có chương trình liệt kê các dãy nhị phân độ dài n bằng thuật toán quay lui như sau:

```
void try(int i) {
    for (int v = 1; v <= n; ++v) {
        if (c[v]) { //giá trị v đang tự do
            x[i] = v;
            if (i == k) // Nếu đã chọn đến bước thứ k thì chỉ cần in ra
                print(x);
            else {
                c[v] = false;  // đánh dấu giá trị v đã được lựa chọn
                try(i+1);
                c[v] = true;   // bỏ đánh dấu, giá trị v trở thành tự do vì x[i] sắp được thử
            }
        }
    }
}
```

5. Liệt kê các hoán vị (bài toán số 1)

Bài toán liệt kê các hoán vị chính là bài toán liệt kê các chỉnh hợp lặp chập k với k = n. Do đó, việc thực hiện không khác gì phần 4. Do k = n nên ta thay điều kiện i == k bằng i ==n là được.

```
void try(int i) {
    for (int v = 1; v <= n; ++v) {
        if (c[v]) { //giá trị v đang tự do
```

```
            x[i] = v;
            if (i == n) // Nếu đã chọn đến bước thứ n thì chỉ cần in ra
                print(x);
            else {
                c[v] = false;  // đánh dấu giá trị v đã được lựa chọn
                try(i+1);
                c[v] = true;   // bỏ đánh dấu, giá trị v trở thành tự do vì x[i] sắp được thử
            }
        }
    }
}
```

6. Discussion question

1. Khi nào thì cần ghi nhận việc phần tử thứ i được gán giá trị v, khi nào không?
2. So sánh thứ tự liệt kê của backtracking và tuần tự (nằm trong 2.1 Exhaustive search (Thuật toán vét cạn)) cho bài toán liệt kê các chỉnh hợp không lặp chập k.
3. Sự khác nhau cơ bản trong cách sử dụng của cách liệt kê bằng backtracking và tuần tự là gì?
4. Thay đổi cách cài đặt các chương trình trên để thuật toán dừng khi liệt kê đến một cấu hình định trước.

No labels