acm International Collegiate Programming Contest

IBM. event sponsor

# NCKU Programming Contest Training Course
# Course 7
# 2013/02/20

http://myweb.ncku.edu.tw/~p76014143/Course7.rar

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

made by electron & free999

# Outline

backtracking

↓

prune technique

↓

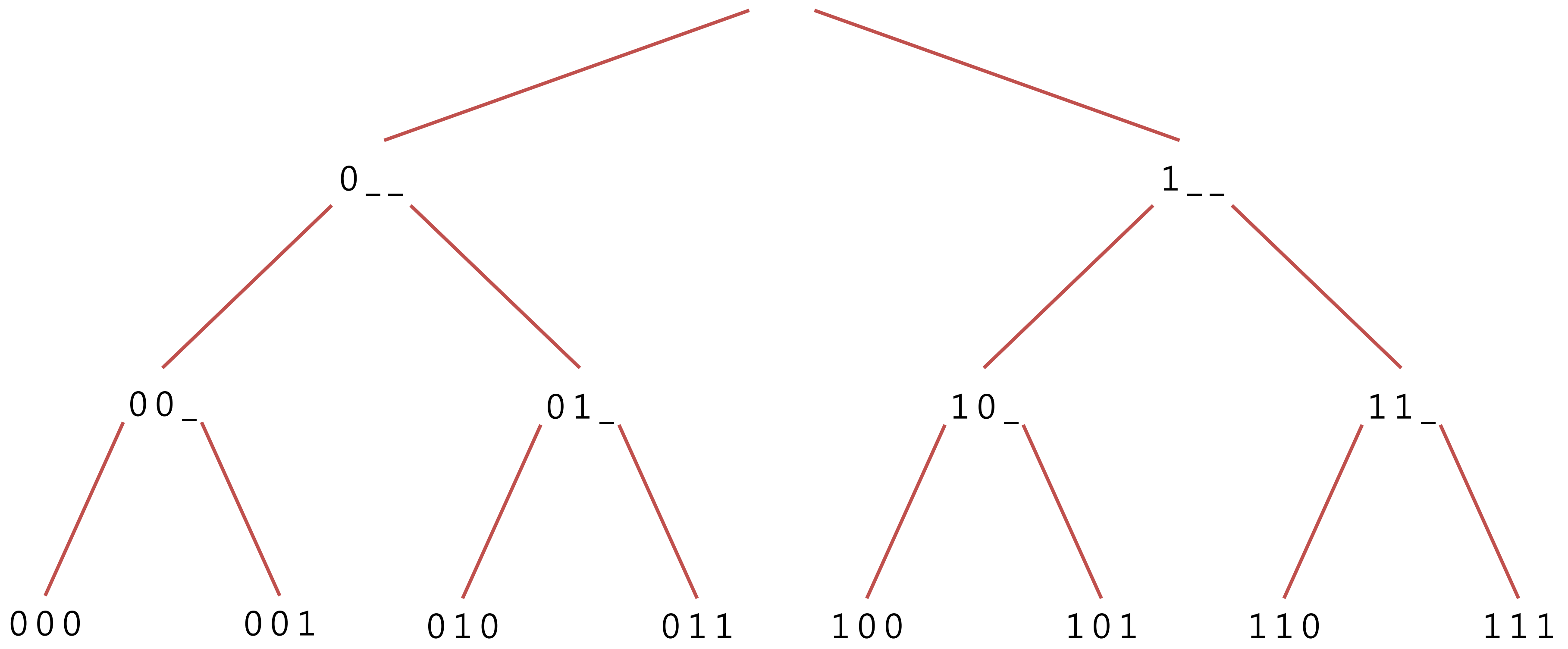
more example

# Backtracking

- Backtracking
  - a general algorithm for finding all (or some) solutions to some computational problem, that incrementally builds candidates to the solutions, and abandons each partial candidate *c* ("backtracks") as soon as it determines that *c* cannot possibly be completed to a valid solution
- Design Method
  - DFS based recursion
  - constraint setting

# Backtracking

- Example of enumerating 3-bits gray codes



# Backtracking

- General pseudo code

```
backtrack( [v1,…,vn] )    // [v1,…,vn] is multiple dimension vector
{
   /* a solution candidate */
   if ( [v1,…,vn] is well-generated ){
      if ( [v1,…,vn] is a solution ) process solution;
      return;
   }

   /* set constraints and recursion */
   for ( x = possible values of vn+1 ){
      set up constraints;
      backtrack( [v1,…,vn, x] );
      back up the constraints
   }
}
call backtrack( [] );   // call function
```

# Backtracking

- Permutation
  - a permutation of a finite set S is a bi-jective map from S to itself; in other words, the any ordering of its element in a list
  - S={1, 2, 3}, the permutation are as follows:
    - {1, 2, 3} {1, 3, 2} {2, 1, 3} {2, 3, 1}, {3, 1, 2}, {3, 2, 1}
- Algorithm
  - backtracking

# Backtracking

- Pseudo code

```
int solution[MAX];      // a candidate
bool used[MAX];         // constraint
void permutation(int k, int n)                  //the kth dimension
{
        if (k == n) // it's a solution
        {
                for (int i=0; i<n; i++)
                cout << solution[i] << " "; cout << endl;
        }
        else {
                for (int i=0; i<n; i++)   // try to enumerate all possible number
                        if (! used[i])
                        {
                                used[i] = true; // set constraint
                                solution[k] = i; // set solution
                                permutation(k+1, n); // recursive
                                used[i] = false; // back up the constraint
                        }
        }
}
```

## Backtracking

- Subset enumeration
  - s is the subset of S means that all elements in s belong to S
  - S={1, 2, 3}
    - s can be {1}, {2, 3}, {1, 3}, ….
    - total $2^n$
- Algorithm
  - backtracking

# Backtracking

- Pseudo code

```
void backtrack(int n)   // n is the dimension
{
        // it's a solution
        if (n == 3)
        {
                print_solution();
                return;
        }
         // take n and set constraint
        solution[n] = true;
        backtrack(n+1);
        // back up the constraint and take nothing
        solution[n] = false;
        backtrack(n+1);
}
```

# Example - 1

- uva 441: loto

Sample Input

```
7 1 2 3 4 5 6 7
```

Sample Output

```
1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 6 7
1 2 3 5 6 7
1 2 4 5 6 7
1 3 4 5 6 7
2 3 4 5 6 7
```

# Exercise

- Problem 1: Permutation
  - input: N (1~10)
  - output: All permutation of 1~N lexically
  - example:
    - N=3
    - {1, 2, 3} {1, 3, 2} {2, 1, 3} {2, 3, 1} {3, 1, 2} {3, 2, 1}

- Problem 2: Partition of string
  - input: string (length <=10)
  - output: all possible partition
  - example:
    - string = "123"
    - (123) (1,23) (12,3) (1,2,3)

## Outline

$$\begin{tikzcd}
\text{backtracking} \arrow[d] \\
\text{prune technique} \arrow[d] \\
\text{more example} \arrow[d] \\
\text{Heap}
\end{tikzcd}$$

## Prune Technique

- Prune
  - if can’ t be, return
  - cut the solution search tree
  - also referred to as branch and bound

# Outline

backtracking

↓

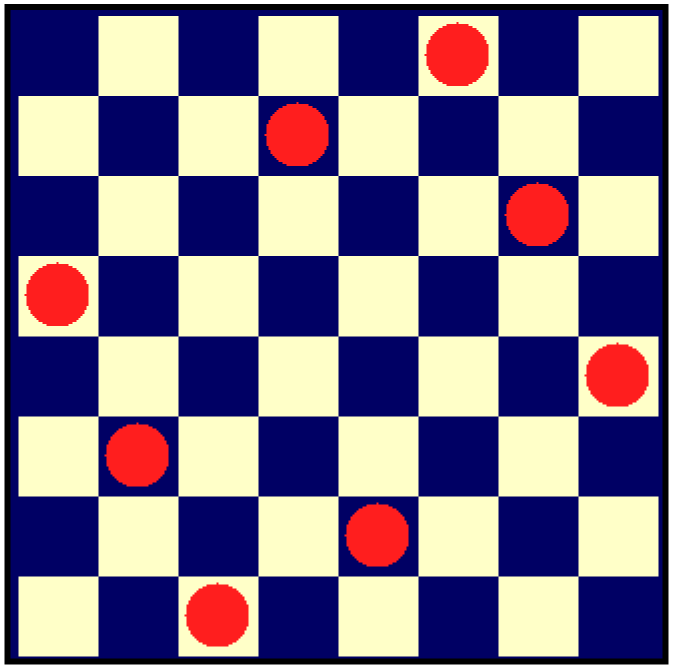
prune technique

↓

more example

# Queen Problem

- 8 Queen Problem (Uva 750)



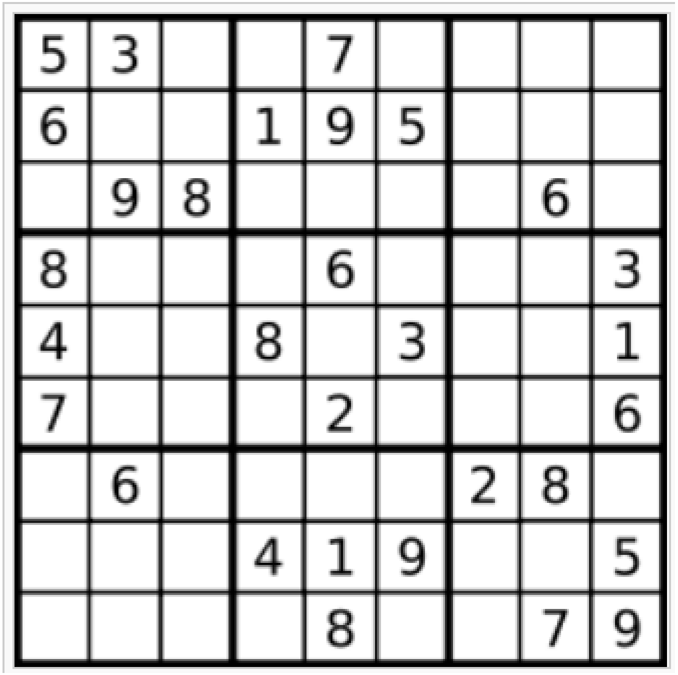
# Queen Problem

- Pseudo code

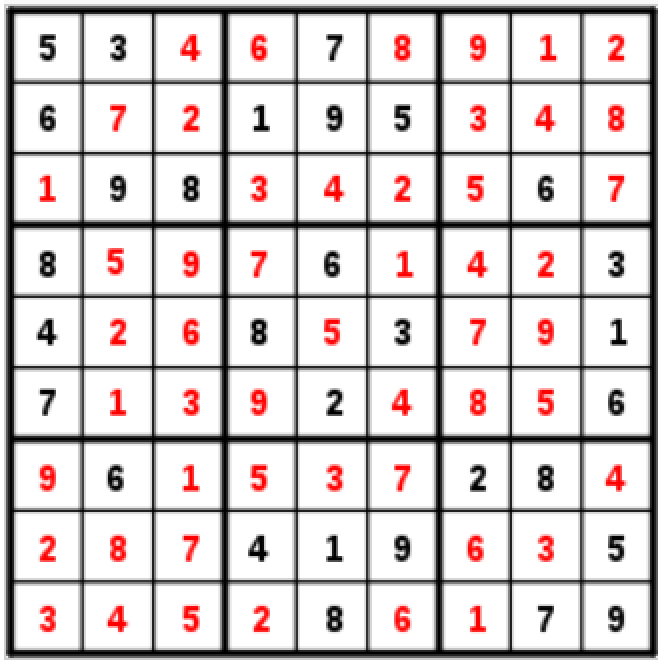
```
void backtrack(int x)   // each row
{
    if (x == 8)             //find a solution
    {
        print_solution();
        return;
    }
    for (int y=0; y<8; ++y)  // try and recursion
    {
        int d1 = (x+y) % 15, d2 = (x-y+15) % 15;
        if (! my[y] && ! md1[d1] && ! md2[d2]) {
            // set up the constraint
            my[y] = md1[d1] = md2[d2] = true;
            solution[x] = y;
            backtrack(x+1);
            // back up the constraint
            my[y] = md1[d1] = md2[d2] = false;
        }
    }
}
```

# Sudoku Problem

- Sudoku Problem (ZJ2d060)



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

# Sudoku Problem

- Pseudo code

```
void backtrack(int x, int y)
{
    if (y == 9) x++, y = 0; // next row
    if (x == 9)             // a solution
    {
        print_solution();
        return;
    }
    // try and recursion
    for (int n=1; n<=9; ++n)
        if (!mx[x][n] && !my[y][n] && !mg[x/3][y/3][n])
        {
            mx[x][n] = my[y][n] = mg[x/3][y/3][n] = true;
            solution[x][y] = n;
            backtrack(x, y+1);
            mx[x][n] = my[y][n] = mg[x/3][y/3][n] = false;
        }
}
```

# Homework 7

- Queen Problem

  Uva 167 750 10513 639 750
- UVA (total 40 problems)
  - 861 10181 10128 10160 10032 10001 704 10270
  - 140 165 193 222 259 291 301 399 435 524 539 565 574 598 628 656 732 10624
- zero judge 2
  - d060(NCPC)

acm International Collegiate Programming Contest

IBM. event sponsor

# Thank You For Attention!

*made by electron & free999*