HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

C BASIC

RECURSIVE BACKGRACKING

ONE LOVE. ONE FUTURE.

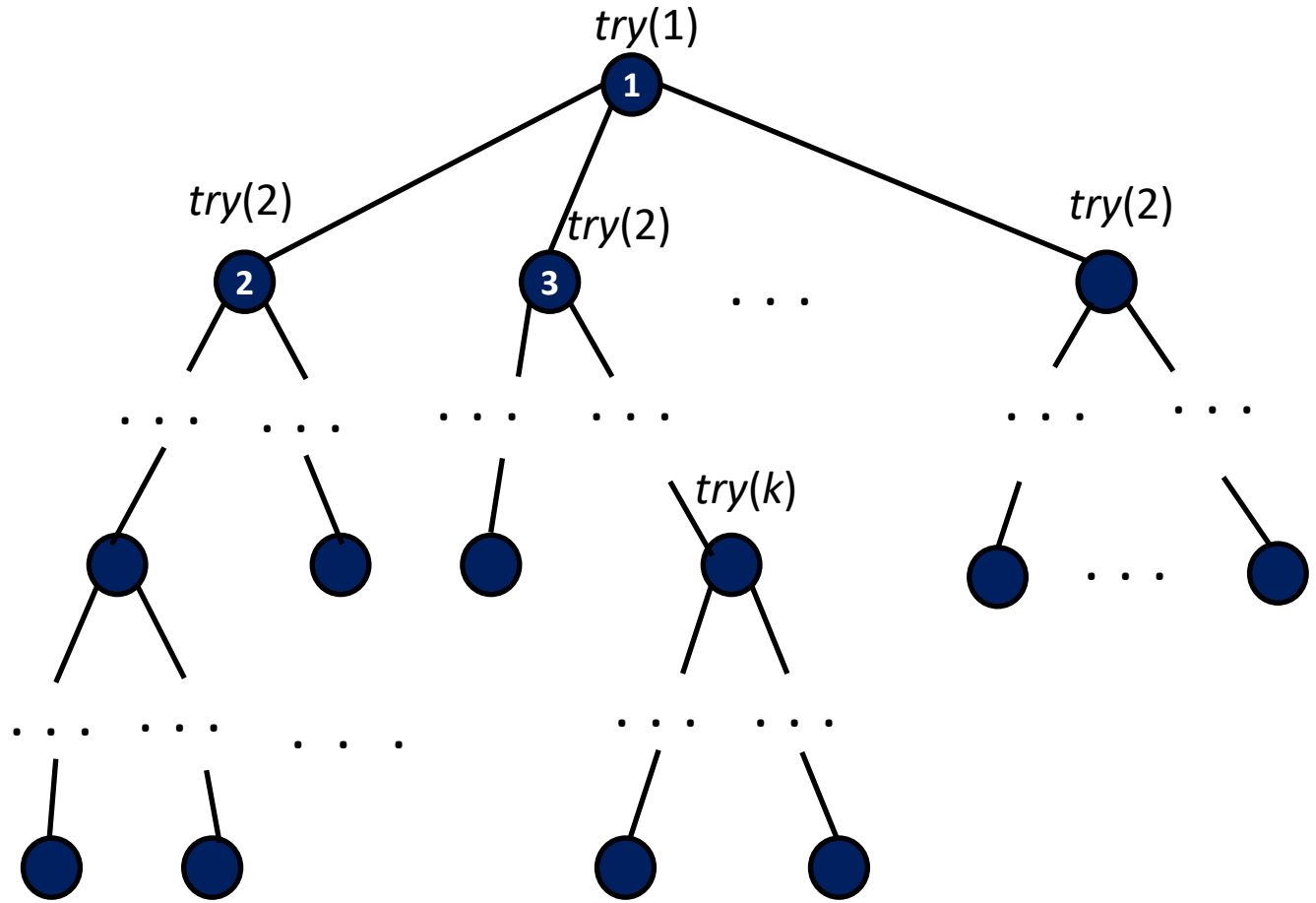
CONTENT

- Recursive backtracking
- Listing a binary string of length n (P.02.05.01)
- Listing a binary string of length n without two adjacent 1 bits (P.02.05.02)
- Listing permutations (P.02.05.03)
- Listing positive integer solutions of linear equations(P.02.05.04)

RECURSIVE BACKTRACKING

- The backtracking algorithm allows us to solve combinatorial enumeration problems and combinatorial optimization problems
- The alternative is modeled by a series of decision variables X1, X2, . . ., Xn
- Need to find for each variable Xi a value from a given discrete set Ai such that
  - The constraints of the problem are satisfied
  - Optimize a given objective function
- Search back and forth
  - Iterate through all variables (e.g. order from X1, X2, . . ., Xn), for each variable Xk
    - Iterate through all possible values assigned to Xk, for each value v
    - Check constraints
    - Assign to Xk
    - If k = n then record one option
    - On the contrary, consider the convolution Xk+1

RECURSIVE BACKTRACKING



Listing problem

```
try(k){ // Try possible values to assign to X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then solution();
        else {
            try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```

LISTING A BINARY STRING OF LENGTH N (P.02.05.01)

- Given a positive integer n. Write a program to list all binary strings of length n in lexicographic order.
- Data
  - Line 1: A positive integer *n* (1 <= *n* <= 20)
- Result
  - Each line a binary string with length *n*

stdin	stdout
3	000 001 010 011 100 101 110 111

LISTING A BINARY STRING OF LENGTH N – PSEUDOCODE

- Representing solutions: $X_1, X_2, \ldots, X_n$
- Function try(*k*):
  - Iterate values of v from 0 to 1
- Function check(*v*, *k*):
  - Always return true

```
check(v, k){
  return true;
}

try(k){
  for v = 0 to 1 do {
     if check(v, k) then {
       X_k = v;
       if k = n then solution();
       else try(k+1);
     }
  }
}
```

LISTING A BINARY STRING OF LENGTH N WITHOUT TWO ADJACENT 1 BITS (P.02.05.02)

- Given a positive integer n. Write a program to list all binary strings of length n (in lexicographic order) that do not contain two adjacent 1 bits.
- Data
  - Line 1: An positive integer *n* (1 <= *n* <= 20)
- Result
  - Each line a bit string of length *n*

stdin	stdout
3	000 001 010 100 101

LISTING A BINARY STRING OF LENGTH N WITHOUT TWO ADJACENT 1 BITS - PSEUDOCODE

- Representing solutions : $X_1, X_2, \ldots, X_n$
- Function try(k):
  - Iterate values of v from 0 to 1
- Function check(v, k):
  - If k = 1 then return true
  - If k > 1
    - If $x[k\text{-}1]$ = 1 và v = 1 then return false
    - Otherwise, return true

```
check(v, k){
  if k = 1 then return true;
  return x[k-1] + v <= 1;
}

try(k){
  for v = 0 to 1 do {
     if check(v, k) then {
       X_k = v;
       if k = n then solution();
       else try(k+1);
     }
  }
}
```

LISTING PERMUTATIONS (P.02.05.03)

- Given a positive integer n. Write a program to list all permutations of 1, 2, . . ., n in lexicographic order
- Data
  - One line with a positive integer n (1 <= *n* <= 10)
- Result
  - Each line a permutation (with SPACE separator for each element)

stdin	stdout
3	1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1

LISTING PERMUTATIONS - PSEUDOCODE

- Representing solutions: $X_1, X_2, \ldots, X_n$
- Marked array:
  - mark[v] = 1: v does appear
  - mark[v] = 0: v does not appear
- Function try(*k*):
  - Iterate values of v from 1 to n
- Function check(*v*, *k*):
  - If mark[v] = 1 then return false
  - Otherwise, return true
- When assigning $X_k = v$:
  - Mark the status mark[*v*] = 1

```
check(v, k){
  if mark[v] = 1 then return false;
  else return true;
}

try(k){
  for v = 1 to n do {
    if check(v, k) then {
      X_k = v;
      mark[v] = 1; // update status
      if k = n then solution();
      else try(k+1);
      mark[v] = 0; // recover when backtracking
    }
  }
}
```

LISTING POSITIVE INTEGER SOLUTIONS OF LINEAR EQUATIONS (P.02.05.04)

- Given positive integers n and M, write a program to list all sets X1, X2, ..., Xn (in lexicographic order) such that: $X_1 + X_2 + \ldots + X_n = M$
- Data
  - Line 1: Two positive integers *n* and *M* (2 <= *n* <= 10, 1 <= *M* <= 20)
- Result
  - Each line a set of values of $X_1, X_2, \ldots, X_n$ (with SPACE separator)

stdin	stdout
3 5	1 1 3 1 2 2 1 3 1 2 1 2 2 2 1 3 1 1

LISTING POSITIVE INTEGER SOLUTIONS OF LINEAR EQUATIONS - PSEUDOCODE

- Representing solutions: $X_1, X_2, \ldots, X_n$
- Intermediate variable T: sum of the assigned values
- Function try(k):
  - $X_1 + X_2 + \ldots + X_{k-1} + X_k + X_{k+1} + \ldots + X_n = M$
  - $T = X_1 + X_2 + \ldots + X_{k-1} \rightarrow X_k \leq M - T - n + k$ (do $X_{k+1}, \ldots, X_n \geq 1$)
  - Iterate values of v from 1 to $M - T - n + k$
- Function check(v, k):
  - If $k < n$ then return true
  - Otherwise
    - if $T + v = M$ then return true, otherwise return false
- When assigning $X_k = v$:
  - Update $T = T + v$

```
check(v, k){
  if k < n then return true;
  else return (T + v = M);
}

try(k){
  for v = 1 to M – T – n + k do {
    if check(v, k) then {
      Xk = v;
      T = T + v; // update status
      if k = n then solution();
      else try(k+1);
      T = T - v; // recover when backtracking
    }
  }
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !