**Algorithm Design Methods**

|  |  |
| --- | --- |
| A person in a suit standing outside  Description automatically generated | Lecturer  PGS TSKH Nguyễn Xuân Huy  Viện Công nghệ Thông tin  Viện Hàn lâm Khoa học  và Công nghệ Việt nam  [nxhuy564@gmail.com](mailto:nxhuy564@gmail.com)  0903203800  Connection: Zalo |

**Contents**

[Tài liệu tham khảo 3](#_Toc171324157)

[Lesson 1 3](#_Toc171324158)

[How many approaches to solve a problem? 3](#_Toc171324159)

[Các quân Hậu Queens(n) 3](#_Toc171324160)

[Next 5](#_Toc171324161)

[NextPerm 5](#_Toc171324162)

[Queens using NextPerm 7](#_Toc171324163)

[BackTracking 9](#_Toc171324164)

[Sudoku 9](#_Toc171324165)

[More of Next 9](#_Toc171324166)

[Max Value 9](#_Toc171324167)

[PlusMinusSet 9](#_Toc171324168)

[ABString 10](#_Toc171324169)

[Next Comb 12](#_Toc171324170)

[*Output* 13](#_Toc171324171)

[Phương án 2 13](#_Toc171324172)

[Phần tử mới 14](#_Toc171324173)

[Tổ hợp sát sau 15](#_Toc171324174)

## Tài liệu tham khảo

1. Bùi Việt Hà, *Nhập môn thuật toán,* Nhà xuất bản Đại học Quốc gia Hà Nội, 2021.
2. Nguyễn Xuân Huy, *Sáng tạo trong thuật toán và lập trình, T5, T6, T7, T8,* Nhà xuất bản Thông tin và Truyền thông*,* 2010 - 2022.
3. Nguyễn Xuân Huy, *Sáng tạo trong thuật toán và lập trình với Python* (tuyển các bài toán tin nâng cao cho học sinh và sinh viên giỏi), Nhà xuất bản Đại học Quốc gia Hà Nội, 2021.
4. Nguyễn Thanh Tùng, *Ngôn ngữ lập trình Python* Athena XIII (pdf)
5. [Crandall Richard](https://en.wikipedia.org/wiki/Richard_Crandall), [Pomerance Carl](https://en.wikipedia.org/wiki/Carl_Pomerance), [*Prime Numbers: A Computational Perspective*](https://books.google.com/books?id=RbEz-_D7sAUC&pg=PA121) (2nd ed.), Springer, p. 121, 2005, [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [9780387252827](https://en.wikipedia.org/wiki/Special:BookSources/9780387252827).
6. Ellis Horowitz, Sartal Sahni, *Fundamentals of Computer Algorithms*, PITMAN, 1978.
7. Donald E. Knuth, *The Art of ComputerPprogramming*, Addison Wesley PC, T1,2,3, 1974.
8. Kenneth H. Rosen, *Elementary Number Theory And Its Applications* (3rd ed.), Addison Wesley PC, 1992.
9. <https://stats.ioinformatics.org/>
10. <https://icpc.global/regionals/abouticpc>

# Lesson 1

**Clean Room: IBM 1 / Test**

## How many approaches to solve a problem?

**range, [ ], function, self function,**

Wing 101

Wing Python

range(a, b, d) a..b step d

range(1,10) d = 1 1..9

range(10) d = 1 0..9

range(10,1,-1) 10..2

s = list(range(10, 4, -1))

[10, 9, 8, 7, 6, 5]

s = list(range(10))

s = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

s[a:b:d] = s[a..b-1] d > 0, d < 0

d > 0: a🡪|b

d < 0: b|🡨a

default a = 0|len(s)-1, d = 1, b = len{s)-1

s[::-1] return reversed s

s[0], s[-1]

...

Hàm tự thân: itself update F(s): update s s.sort

Hàm tạo mới: w = F(s) sorted(s)

## Các quân Hậu Queens(n)

Quân Hậu trên bàn cờ Vua có thể diệt quân đối phương theo hàng, theo cột chứa nó hoặc theo đường chéo của hình vuông nhận nó làm đỉnh.

* Tìm một cách đặt N quân Hậu trên bàn cờ Vua kích thước N× N sao cho không quân nào diệt được quân nào.
* Tìm mọi cách đặt N quân Hậu theo điều kiện trên.

Hiển thị kết quả trên màn hình.

Ví dụ

A screenshot of a game

Description automatically generated with medium confidence

Chart

Description automatically generated

v = [1,5,8,6,3,7,2,4]

với ý nghĩa:

Đặt Hậu 1 tại dòng 1

Đặt Hậu 2 tại dòng 5

Đặt Hậu 3 tại dòng 8

Đặt Hậu 4 tại dòng 6

Đặt Hậu 5 tại dòng 3

Đặt Hậu 6 tại dòng 7

Đặt Hậu 7 tại dòng 2

Đặt Hậu 8 tại dòng 4

**n ≤ 8: vét cạn**

n = 4

1234

1243

1324

1342

…

4321

x = **1432**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** |  |  |  |
|  |  |  | **2** |
|  |  | **3** |  |
|  | **4** |  |  |

Algorithm

Init x = [1,...,n]

for each permuation x:

if Sat(x): print(x)

**Permutation?**

**Ver 1. Using library**

# Queens. Ver 1: using library

from itertools import permutations # lib, module

from time import time

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def Queens(n):

print('Queen of ', n, ': ')

if n >= 10:

print('Large Chess Board')

return

if n < 1: print(); return

if n == 1: print('(1)'); return

a = list(range(1,n+1)) # a = [1,..,n]

p = permutations(a) # lib

c = 0

for x in p:

if Sat(x):

c += 1

print(c, ':', x)

print('Total ', c)

# x = [1,4,3,2]

# Sat(x) ?

def Sat(x):

for i in range(1,len(x)):

for j in range(i):

if x[i] == x[j] or i-j == abs(x[i]-x[j]):

return False

return True

# APPLICATION

for n in range(15):

t = time()

Queens(n)

print('Total time of Queens ', n, ':', time()-t)

Go()

print(' T h e E n d')

**Ver 2. Myself**

## Next

Given a posituve integer x. Swap some digits in x to get the next

nearest number y of x. max len(x) = 200

Example

x = 134763

y = 136347

x < y, no z: x < z < y

Algorithm Next: 2 Phases

A diagram of a diagram

Description automatically generatedPhase 1. Duyet nguoc tim diem gay i

Neu khong co diem gay: return empty

Phase 2. Duyet nguoc tim diem vuot j

Swap i, j

rev x[i+1..]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **3** | **4** | **7** | **6** | **3** |
| diem gay |  |  | i |  |  |  |
| diem vuot |  |  |  |  | j |  |
| swap | 1 | 3 | **6** | 7 | **4** | 3 |
| final | 1 | 3 | 6 | 3 | 4 | 7 |

## NextPerm

# NextPerm: Hoán vị kế tiếp

def Go(msg = ' ? '):

if input(msg) == '.':

exit(0)

def NextPerm(s): # Self update s -> s

if s == []: return False

n = len(s)

i, j = 0, 0

# Pha 1: duyệt ngược tìm điểm gãy i

# 01234

for i in range(n-2, -2, -1):

if s[i] < s[i+1]: break

#print('\n diem gay: ', i)

if i < 0: s.clear(); return False # không có điểm gãy

# pha 2: duyệt ngược tìm điểm vượt j (luôn có)

for j in range(n-1, i, -1):

if s[j] > s[i]: break

# ----i-----j-----

# đổi chỗ s[i], s[j]

s[i], s[j] = s[j], s[i]

# lật lại đoạn s[i+1:]

i , j = i+1, n-1

while i < j:

s[i], s[j] = s[j], s[i]

i, j = i+1, j-1

return True

def Test1():

print('Test No 1')

s = '134763'

print(s, end = ' -> ')

s = list(map(int,s))

if NextPerm(s):

print(''.join([str(c) for c in s]))

else: print('Empty')

s = list(map(int,'7654321'))

print(s, end = ' -> ')

if NextPerm(s):

print(''.join(map(str, s)))

else: print('Empty')

def Print(d, msg = ''):

if msg != '': print(msg)

c = 0

for x in d: c += 1; print(c,'.', x)

def Test2():

print('Test No 2')

s = [1,2,3,4]

print(' All Permutations of ', s)

Print(AllPerm(s))

def AllPerm(s):

p = []

while len(s) > 0:

p.append(s.copy())

if not NextPerm(s): break

return p

def GetNextPerm(s): # return w, s bao luu

w = s.copy()

if NextPerm(w): return w

else: return []

def GetAllPerm(s):

p = []

if s == []: return p

while len(s) > 0:

p.append(s)

s = GetNextPerm(s)

return p

def Test3():

print('Test No 3')

s = [1,2,3]

print(' All Permutations of ', s)

Print(GetAllPerm(s))

# APPLICATION

Test1()

Test2()

Test3()

print(' T h e E n d')

## Queens using NextPerm

# Queens using (My)NextPerm

def Go(msg = ' ? '):

if input(msg) == '.':

exit(0)

def NextPerm(s): # Self update s -> s

if s == []: return False

n = len(s)

i, j = 0, 0

# Pha 1: duyệt ngược tìm điểm gãy i

# 01234

for i in range(n-2, -2, -1):

if s[i] < s[i+1]: break

#print('\n diem gay: ', i)

if i < 0: s.clear(); return False # không có điểm gãy

# pha 2: duyệt ngược tìm điểm vượt j (luôn có)

for j in range(n-1, i, -1):

if s[j] > s[i]: break

# ----i-----j-----

# đổi chỗ s[i], s[j]

s[i], s[j] = s[j], s[i]

# lật lại đoạn s[i+1:]

i , j = i+1, n-1

while i < j:

s[i], s[j] = s[j], s[i]

i, j = i+1, j-1

return True

def AllPerm(s):

p = []

while len(s) > 0:

p.append(s.copy())

if not NextPerm(s): break

return p

def GetNextPerm(s): # return w, s bao luu

w = s.copy()

if NextPerm(w): return w

else: return []

def GetAllPerm(s):

p = []

if s == []: return p

while len(s) > 0:

p.append(s)

s = GetNextPerm(s)

return p

def Print(d, msg = ''):

print(msg)

c = 0

for x in d:

c += 1

print(c,'.', x)

print('Total', c)

# sn = 1: one solution

# sn != 1: all solutions

def Queens(n, sn = 1):

print('Queens of ', n, end=': ')

d = []

if n > 8:

print('Large of chessboard'); return d

if n < 1: return d

if n == 1: return [1]

if n < 4: return d

# n >= 4 having solutions

s = list(range(1,n+1))

if sn == 1: return Q1(s)

else: return QM(s)

# Hau tren hang i va j chieu nhau

def Sat(s): # s = [-j--i--]

for i in range(1, len(s)):

for j in range(i):

if s[i] == s[j] or i-j == abs(s[i]-s[j]):

return False

return True

def Q1(s): # find one solution

while not Sat(s):

if not NextPerm(s): return []

return s

def QM(s): # find all solution

return [x for x in AllPerm(s) if Sat(x)]

def Test1():

print('One solution:')

for n in range(-1, 10):

print(Queens(n))

Go()

def Test2(): # all solution

print('All solutions:')

for n in range(4, 10):

d = Queens(n, 2)

Print(d)

Go()

# APPLICATION

Test1()

Test2()

print(' T h e E n d')

## BackTracking

## Sudoku

## More of Next

### Max Value

Cho dãy số nguyên a[1..n]. Hãy đặt n dấu + hoặc – vào trước mỗi số để thu được biểu thức có giá trị lớn nhất.

|  |  |  |
| --- | --- | --- |
| VALUE.INP | OUTPUT | *Dữ liệu vào: Text file VALUE.INP*  *n và dãy số a, 2 ≤ n ≤ 1000.*  *Dữ liệu ra: dãy phép toán.* -+-++  *Ta có:* -(-1)+2-(-5)+7+8 = 23 |
| 5 | -+-++ |
| -1 2 -5 7 8 |  |

Algorithm

Giá trị max của biểu thức chính là tổng các giá trị tuyệt đối của các số hạng, do đó ta sẽ đặt các dấu – vào trước các số hạng âm, còn lại là các số hạng dương kèm dấu +.

Program

# Value

FN = 'VALUE.INP'

def Go(msg = ' ? '):

if input(msg) == '.':

exit(0)

def Value():

with open(FN) as f:

n = int(f.readline())

print(n)

d = list(map(int,f.readline().split()))

print(d)

val = 0

for x in d:

if x < 0: print('-', end = '')

else: print('+', end = '')

val += abs(x)

print('\n Max val = ', val)

Value()

print(" T H E E N D .")

Output

5

[-1, 2, -5, 7, 8]

-+-++

Max val = 23

T H E E N D .

### PlusMinusSet

Cho dãy số nguyên a[1..n]. Hãy đặt n dấu + hoặc – vào đầu mỗi số để thu được biểu thức có giá trị v cho trước.

|  |  |  |
| --- | --- | --- |
| PMVALUE.INP | OUTPUT | *Dữ liệu vào: Text file PMVALUE.INP*  *n v*  *dãy số a, 2 ≤ n ≤ 1000.*  *Dữ liệu ra: dãy phép toán.* --+++  *Ta có: -(-1) - 2 + (-5) + 7 + 8 = 9* |
| 5 9 | --+++ |
| -1 2 -5 7 8 |  |

Algorithm

Giá trị max của biểu thức chính là tổng các giá trị tuyệt đối của các số hạng, do đó ta sẽ đặt các dấu – vào trước các số hạng âm, còn lại là các số hạng dương kèm dấu +.

Program

# PlusMinusValue

FN = 'PMVALUE.INP'

def Go(msg = ' ? '):

if input(msg) == '.':

exit(0)

def Input():

global n, v, d

with open(FN) as f:

dd = list(map(int,f.read().split()))

n, v = dd[0], dd[1]

d = dd[2:]

# chuyen x thanh dau +-

# tinh val

def Val(x):

b = bin(x)[2:].zfill(n)

val = 0

for i in range(n):

val += d[i] if b[i] == '1' else -d[i]

return ToPM(b) if val == v else ''

def ToPM(b):

return ''.join(map(lambda c:'+'if c=='1' else'-',b))

def Find():

# n dau +-: 2^n

maxn = 2\*\*n

# print(maxn)

for x in range(maxn):

s = Val(x)

if s != '': return s

return ''

def Run():

Input()

print('Input:', n, v, d)

print('Result', Find())

# APPLICATION

Run()

print(" T H E E N D .")

Output

[evaluate a.py]

Input: 5 9 [-1, 2, -5, 7, 8]

Result --+++

T H E E N D .

### ABString

Một xâu kí tự s được gọi là ABString độ dài n nếu mọi kí tự trong S đều có giá trị A hoặc B.

Ví dụ xâu S = “ABABABAB” là một xâu ABString độ dài 8.

Với mỗi cặp giá trị n và k, hãy liệt kê các xâu ABString dài n và chứa duy nhất một dãy k kí tợ A

liên tiếp.

n ≤ 30

Ví dụ

|  |  |
| --- | --- |
| AB.INP | OUTPUT |
| 5 3 | 5  AAABA  AAABB  ABAAA  BAAAB  BBAAA |

Algorithm

Vét cạn các ABString s len = n. Nếu s chứa duy nhất string(k,A) thì lấy.

n = 5, k = 3

01101001110, samle = 111

BAABABBAAAB BBB

01101001110

111 &

------------

110

0110100111

111 &

------------

111 : 1

011010011

111 &

------------

011

...

Given n, k

sample

for x = minn..maxn

if Sat(x, sample): get

**Sat?**

c = 0

while x > 0:

if x & sample == sample: ++c

x >>= 1

return c == 1

Program

# ABString

def Go(msg = ' ? '):

if input(msg) == '.':

exit(0)

def ToAB(x, n):

b = bin(x)[2:].zfill(n)

return ''.join(map(lambda c:'A'if c=='1' else'B',b))

def ABString(n, k):

global sample

d = []

sample = 2\*\*k-1

print('sample: ', sample)

for x in range(2\*\*n):

if Sat(x, n): d.append(x)

print(len(d))

for x in d: print(ToAB(x, n))

def Sat(x, n):

c = 0 # counter

for i in range(n):

if x & sample == sample: c += 1

x >>= 1

return c == 1

# APPLICATION

ABString(5, 3)

print(" T H E E N D .")

Output

sample: 7

5

BBAAA

BAAAB

ABAAA

AAABB

AAABA

T H E E N D .

## Next Comb

Tổ hợp / Combinators

C(5, 3) abc abd abe acd ace ade bcd bde cde

C(n, k) =

C(n,k) ?

Next Comb ?

Các tính chất của hàm hai ngôi C(n,k)

* C(n,0) = 1 (theo quy ước 0! = 1! = 1)
* C(n,1) = n
* Tính đối xứng C(n, k) = C(n, n-k)
* C(n, k) = C(n-1, k-1) + C(n-1, k), n ≥ 2, 0 < k < n
* C(n,k) là hệ số trong dạng khai triển của nhị thức Newton
* Ta biết, trong dạng khai triển trên, hai hệ số hai đầu là . Các hệ số còn lại được gọi là *các hệ số trong*, cụ thể là các hệ số với 0 < k < n. Số tự nhiên *n* là nguyên tó khi và chỉ khi các hệ số trong của dạng khai triển nhị thức Newton đều là bội của n

Ví dụ n = 5 cho ta c[1..4] = (5, 10, 10, 5). Vậy 5 là số nguyên tố.

Có hai phương pháp tính hàm C(n,k).

Phương pháp lặp

Vận dụng tính đối xứng để giảm bớt số lần lặp: nếu k > n/2 thì thay vì tính C(n,k), ta tính C(n, n-k).

# Tinh C(n,k) bang phuong phap lap

def C(n,k):

if k+k > n: k = n-k

c, n1 = 1, n+1

for i in range(1,k+1): c = c\*(n1-i)//i

return c

Phương pháp đệ quy

def RC(n, k):

return 1 if k == 0 else RC(n,k-1)\*(n-k+1)//k

# Tinh C(n,k) bang phuong phap de quy

def C2(n,k):

return RC(n,k) if k + k <= n else RC(n, n-k)

Độ phức tạp

Tính hàm C(n, k) cần k lần thực hiện các phép nhân và chia.

Chương trình sau đây tính các hệ số của của dạng khai triển nhị thức Newton Hãy quan sát những dòng là số nguyên tố.

Chương trình

# To hop

# Cac he so trong (a+b)^n

def Go():

if input(' ? ') == '.':

exit(0)

# Tinh C(n,k) bang phuong phap lap

def C(n,k):

if k+k > n:

k = n-k

c, n1 = 1, n+1

for i in range(1,k+1):

c = c\*(n1-i) // i

return c

def RC(n, k):

return 1 if k == 0 else RC(n,k-1)\*(n-k+1)//k

# Tinh C(n,k) bang phuong phap de quy

def C2(n,k):

return RC(n,k) if k + k <= n else RC(n, n-k)

def Run():

for n in range(1,11):

print('\n n = ', n, ':', end = ' ')

for k in range(n+1):

print(C(n,k), end = ' ')

#print(C2(n,k), end = ' ')

Run()

print('\n T h e E n d')

Output

n = 1: 1 1

n = 2: 1 2 1

n = 3: 1 3 3 1

n = 4: 1 4 6 4 1

n = 5: 1 5 10 10 5 1

n = 6: 1 6 15 20 15 6 1

n = 7: 1 7 21 35 35 21 7 1

n = 8: 1 8 28 56 70 56 28 8 1

n = 9: 1 9 36 84 126 126 84 36 9 1

n = 10: 1 10 45 120 210 252 210 120 45 10 1

T h e E n d

### Phương án 2

Vận dụng tính đối xứng của dãy hệ số c(i) ta có thể tính một nửa số lượng sau đó lấy đối xứng.

Program

# To hop

# Cac he so trong (a+b)^n

# Phuong an 2: van dung tinh doi xung

def Go():

if input(' ? ') == '.': exit(0)

#Tinh C(n,k) bang phuong phap de quy

def C(n,k):

return RC(n,k) if k + k <= n else RC(n, n-k)

def RC(n, k):

return 1 if k == 0 else RC(n,k-1)\*(n-k+1)//k

# d = cac he so c(n,k), k = 0..n

def Row(n):

d = [1] + [0]\*(n-1) + [1] # d = [1,1,0...,0,1]

# len(d) = n + 1

for k in range(1, n//2 + 1): d[k] = d[n-k] = C(n,k)

return d

def Run():

for n in range(1,11):

print('n = ', n, ':', Row(n))

Run()

print('\n T h e E n d')

Chú thích

* Trong Python có hai thư viện math và itertoos cung cấp các hàm về hoán vị và tổ hợp. Bạn quan sát chương trình dưới đây để biết cách sử dụng các hàm có sẵn. # To hop chap 3 cua 5 phan tu

# Cac he so trong (a+b)^n

from math import comb

from itertools import combinations

def Go():

if input(' ? ') == '.': exit(0)

# d = cac he so c(n,k), k = 0..n

def Row(n): return [comb(n,k) for k in range(n+1)]

def Run1():

# Cac he so cua nhi thuc Newton (a+b)^n, n = 1..10

for n in range(1,11):

print('n = ', n, ':', Row(n))

def Run2():

# To hop chap 3 cua 5 phan tu 1..5

comb = combinations([1, 2, 3, 4, 5], 3)

for x in comb: print(x)

Run1()

Run2()

print('\n T h e E n d')

NextComb(n, c)

### Phần tử mới

*Biết rằng các tổ hợp chặp k của n phần tử A,B,C...được sắp tăng theo thứ tự từ điển. Biết một tổ hợp c trong dãy.*

*Hãy cho biết tổ hợp sát sau của tổ hợp này có bao nhiêu phần tử mới được thêm vào.*

Ví dụ

*n =* 5*, k =* 3*. Có 10 tổ hợp được xếp tăng dần như sau:*

ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE, CDE.

Sau tổ hợp ACD là tổ hợp ACE, do đó số phần tử mới là 1 (E).

Sau tổ hợp ADE là tổ hợp BCD, do đó số phần tử mới là 2 (B, C).

Sau tổ hợp CDE không có tổ hợp nào

## Tổ hợp sát sau

Algorithm

Thuật toán tìm tổ hợp sát sau của tổ hợp chặp k của n phần tử gồm hai pha:

Pha thứ nhất: Tìm điểm gãy

Duyệt ngược tổ hợp từ c[k-1] đến c[0] tìm điểm i thỏa điều kiện c[i] không nhận giá trị giảm đều từ n trở về trước. Điểm i được gọi là *điểm gãy*.

Nếu i < 0 chứng tỏ không có tổ hợp tiêp theo.

Nếu i ≥ 0 ta chuyển qua pha thứ hai

Pha thứ hai: Rải đều

Rải đều các thành phần từ c[i..k-1] các giá trị tăng đều từ c[i]+1 trở đi.

Ví dụ. Tìm tổ hợp sát sau của tổ hợp [2,3,6,7]? Ta có:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n = 7, k = 4 | | | | |
| c[] | *0* | *1* | *2* | *3* |
| *tổ hợp* | 2 | 3 | 6 | 7 |
| *điểm gãy i = 1* |  | i |  |  |
| *rải đều* | 2 | **4** | **5** | **6** |
| NextComb [2,3,6,7] = [2,4,5,6] | | | | |

# NextComb: Tổ hợp sát sau

# Tập nền: 1..n

def Go(msg = ' ? '):

if input(msg) == '.':

exit(0)

# Tổ hợp sát sau của tổ hợp comb

# tự cập nhật comb

def NextComb(n, comb):

# input comb = C(n,k)

# Pha 1: tìm điểm gãy

k = len(comb)

sample = list(range(n+1))[-k:]

#print(comb, sample)

if sample == comb: return False

i = 0

for i in range(k-1,-1,-1):

if comb[i] != sample[i]: break

# Pha 2: rải đều

val = comb[i]

for j in range(i, k):

val += 1

comb[j] = val

return True

def AllComb(n, k):

d = []

comb = list(range(1,k+1)) # Init c = [1..k]

while True:

#print(comb.copy());Go()

d.append(comb.copy())

if not NextComb(n, comb): break

return d

# Tổ hợp sát sau của tổ hợp comb

def GetNextComb(n, comb):

c = comb.copy()

if NextComb(n, c): return c

return []

def AllComb2(n, k):

d = []

comb = list(range(1,k+1)) # Init c = [1..k]

while len(comb) > 0:

d.append(comb)

comb = GetNextComb(n, comb)

return d

def Print(d, msg = ''):

print(msg)

c = 0

for x in d:

c += 1

print(c, '.', x)

print('Total ', c)

def NewElem(n, s):

c = list(map(lambda x: ord(x)-ord('A')+1,s))

# print('Next ', s, c, end = ' ')

if not NextComb(n, c): return 'Empty'

# print(c)

w = ''.join([chr(x + ord('A')-1) for x in c])

return Minus(w,s)

def Minus(w,s): # w-s: các kí tự có trong w, ko có trong s

u = ''

for c in w:

if not c in s: u += c

return u

# APPLICATION

n, k = 6, 4

d = AllComb(n, k)

Print(d, 'Phương án 1:')

v = AllComb2(n, k)

Print(v, 'Phương án 2:')

n = 5

print(NewElem(n, 'ACD'))

print(NewElem(n, 'ADE'))

print(NewElem(n, 'CDE'))

print("\n T H E E N D .")

**News**

1. Small Data: vet

2. Backtracking: so do 1 nghiem, so do mai nghiem

3. bit operation &, |, ^, not, &=, |=, ^=, >> >>=, <<, <<=

1 >> k = 2^k x << k = x // (2^k)

4. join, map, list,....