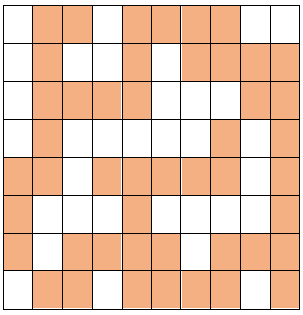
DATA STRUCTURES AND ALGORITHMS
IN C++ (Tom 2)

CONTENTS

Rain

Gặp trời mưa thì có bao nhiêu ô đọng nước?

```
8 10
0110111100
0100101111
0111100011
0100000101
1101111101
1000100001
1011110111
0110111101
```



Bạn phải biết chắc input (giả thiết) và output (kết luận)

Tên file, kiểu dữ liệu

Connect Zone

```
/*************************************
           Rain
*************************************/
#include<iostream>
#include<fstream>
#include<windows.h>

using namespace std;

const char * fn = "RAIN.INP";
const int MN = 200;
const int MN2 = MN*MN;
const char empty = '0';
char a[MN][MN];
int d[MN2+1]; // references
int dt[MN2+1]; // dien tich vung lien thong

int m, n;
int nm;


void Go() {
  cout << " ? ";
```

```
  fflush(stdin);
  if (cin.get() == '.') exit(0);
} // Go()

void Print(char x[MN][MN], int n, int m, const char * msg = "") {
  cout << msg;
  for (int i = 1; i <= n; ++i) {
    cout << "\n ";
    for (int j = 1; j <= m; ++j) {
       cout << x[i][j];
    }   // j
  } // i
} // Print

void Print(int x[], int n, const char * msg = "") {
  cout << msg;
  for (int i = 1; i <= n; ++i) {
       cout << " " << i << ":" << x[i];
       if (i % 15 == 0) cout << endl;
  } // i
} // Print


void Read() {
  ifstream f(fn);
  string s;
  memset(a, empty, sizeof(a));
  f >> n >> m; // n: row, m: column
  cout << " row = " << n << "  column = " << m << endl;
  for (int i = 1; i <= n; ++i) {
    f >> s;
    cout << s << endl;
    for (int j = 0; j < s.length(); ++j) {
      a[i][j+1] = s[j];
    } // for j
  } // for i
  f.close();
} // Read

int Index(int i, int j) {
  return (i-1)*m + (j-1) + 1;
} // Index

void Split(int id, int &i, int &j) {
  i = (id - 1) / m + 1;
  j = (id - 1) % m + 1;
} // split

void Init() {
  nm = n*m;
  cout << "\n n = " << n << " m = " << m << "  nm = " << nm;
  for (int i = 1; i <= nm; ++i)
    d[i] = i;
} // Init

int Find(int x) {
  while (d[x] != x) x = d[x];
  return x;
} // Fond

int Union(int x, int y) {
```

```
  x = Find(x);
  y = Find(y);
  if (x == y) return 0;
  if (x < y) d[y] = x;
  else d[x] = y;
  return 1;
} // Union

// Xac dinh cac vung lien thong
void Connect() {
  int id;
  for (int i = 1; i < n; ++i) {
    for (int j = 1; j < m; ++j) {
      id = Index(i,j);
      // xet cac o ke (i,j) id >
      // ke phai
      if (a[i][j] == a[i][j+1]) // 2 o cung mau
        Union(id, id+1);
      if (a[i][j] == a[i+1][j]) // 2 o cung mau
        Union(id, id + m);
    } // for j
  }  // for i
  // Tro truc tiep
  for (int i = 1; i <= nm; ++i) {
    if (d[i] == i) {
      for (int j = i+1; j <= nm; ++j) {
        if (Find(j) == i)
          d[j] = i;
     } // for j
    }
  } // for i
} // Connect

void Square() {
  // Dien tich cac vung Lien thong
  memset(dt, 0, sizeof(dt));
  // Xet chu vi
  for (int j = 1; j <= m; ++j) {
    if (a[1][j] == empty)
       dt[Find(Index(1,j))] = -1;
    if (a[n][j] == empty)
       dt[Find(Index(n,j))] = -1;
  } // for j
  for (int i = 1; i <= n; ++i) {
    if (a[i][1] == empty)
       dt[Find(Index(i,1))] = -1;
    if (a[i][m] == empty)
       dt[Find(Index(i,m))] = -1;
  } // for i

  // Cac o ben trong
  for (int i = 2; i < n; ++i) {
    for (int j = 2; j < m; ++j) {
    if (a[i][j] == empty) {
      int id = Index(i,j);
      if (d[id] == id && dt[id] == 0) { // head
        ++dt[id];
        // cout << "\n (" << i << "," << j << ")"; Go();
        for (int k = id+1; k <= nm; ++k)
         if (d[k] == id) ++dt[id];
     } // head
```

```
    } // if a
   } // for j
   } // for i
} // Square

void Run2() {
  Read();
  Print(a, n, m, "\n Input: ");
  Init();
  Connect();
  Square();
  // dien tich vung co nuoc
  int s = 0;
  for (int i = 1; i <= nm; ++i)
    if (dt[i] > 0) s += dt[i];
  cout << "\n Dien tich vung chua nuoc " << s;
  int x, y, h;
  while (1) {
    cout << "\n x , y [get a negative number to exit]: ";
   fflush(stdin);
   cin >> x >> y;
   if (x < 0 || y < 0) return;
   h = Find(Index(x,y));
   if (dt[h] < 0) cout << 0;
   else cout << dt[h];
   } // while
} // Run2

main() {
  Run2();
  cout << "\n T H E   E N D";
  return 0;
}
```

Cubic Number

Clean Room (IBM)

```
/************************************************
    cubic.CPP
   Xoa bot cac chu so cua so n de thu duoc so
   lon nhat bang lap phuong cua so khac
   22/08/20 09:32
   4125: 125
   967: -1
************************************************/
#include <iostream>
#include <fstream>
#include <cmath>
#include <string.h>
using namespace std;
typedef long long Long;
const int MN = 1290;
Long num3[MN+1];
string str3[MN+1];
const char * fn = "CUBIC.INP";

void Go() {
  cout << "\n Press dot key [.] to stop: ";
  fflush(stdin);
  if (cin.get() == '.') exit(0);
} // Go

string ToStr(Long x) {
  string s = "";
  while (x != 0) {
    s = (char)((x % 10) + '0') + s;
    x /= 10;
  } // while
  return s;
} // ToStr

// num[i] = i*i*i
void Init() {
  num3[0] = 0;
  Long i3;
  for (int i = 1;  i <= MN; ++i) {
    i3 = i * i * i;
    num3[i] = i3;
    str3[i] = ToStr(i3);
  } // for
} // Int

int Search(Long n) {
  int d = 1, c = MN, m;
  while (d < c) {
```

```
    m = (d+c) / 2;
    if (num3[m] < n) d = m + 1;
   else c = m;
  } // while
  return d;
} // Search

// so khop x va y
bool Match(string x, string y) {
  int i, j, k = -1;
  for (i = 0; i < x.length(); ++i) {
    for (j = k + 1; j < y.length(); ++j) {
      if (y[j] == x[i]) {
       k = j;
       break; // for j
     }
    } // for j
   if (j >= y.length()) return false;
  } // for i
  return true;
} // Match

// return so lap phuong max
Long Find(Long n) {
   string sn = ToStr(n);
   for (int i = Search(n); i > 0; --i)  {
     if (Match(str3[i], sn))
       return num3[i];
   } // for
   return -1;
} // Find

void Test() {
  Init();
  ifstream f(fn);
  int sotest ;
  Long n;
  f >> sotest;
  for (int i = 1; i <= sotest; ++i) {
    f >> n;
    cout << "\n Test no " << i << ". N = " << n;
   cout  << " -> Result = " << Find(n);
  } // for
  f.close();
} // Test;

main() {
  Test();
  cout << "\n T H E   E N D";
  return 0;
}
```

Test data

cubic.inp
7
4125
976
9413192

10007000
238
127731561
438698939

Đáp án

N = 4125: Result = 125

N = 976: Result = -1

N = 9413192: Result = 941192

N = 10007000: Result = 1000000

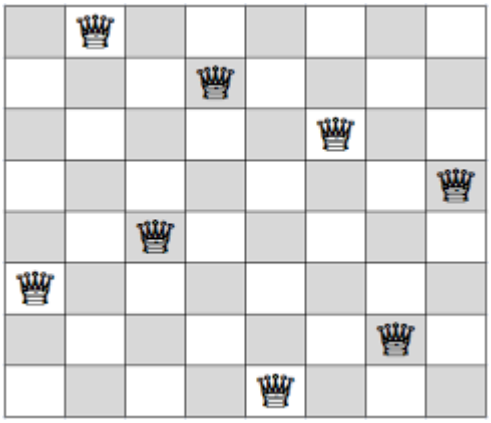
N = 238: Result = 8

N = 127731561 : Result = 1771561

N = 438698939: Result = 3869893

Queens in places

Nếu mỗi ô (i,j) trên bàn cờ có một giá trị không âm thì bố trí các quân hậu ra sao để thu được tổng giá trị lớn nhất.



Thuật toán

Tìm trị của mọi nghiệm và lấy max.

```
/*****************************************
   QP.CPP (Queens in places)

   23/08/20 10:10

*****************************************/
#include <iostream>
#include <ctime>
#include <windows.h>

using namespace std;

const int MN = 100;
const int MN1 = MN + 1;
const char BL = 32;
int a[MN1][MN1];
int pos[MN1];
int posmax[MN1];

void Print(int x[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      if (x[i] < 10) cout << BL;
      cout << BL << x[i];

   } // for
} // Print

void Print(int a[][MN1], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      Print(a[i], d, c, "\n ");
   } // for
} // Print
```

```
void Gen(int n) {
   for (int i = 1; i <= n; ++i) {
      for (int j = 1; j <= n; ++j) {
         a[i][j] = rand() % 100;
      } // for j
   } // for i

} // Gen

int SumVal(int pos[], int n) {
   int s = 0;
   for (int k = 1; k <= n; ++k) {
      s += a[pos[k]][k];
   } // for
   return s;
} // Sum

bool GoodPlace(int k, int i) {
   for (int j = 1; j < k; ++j) {
      if (pos[j] == i || k - j == abs(pos[j] - i))
         return false;
   } // for
   return true;
} // GoodPlace

// Find a place (row i) for Queen k
int Find(int n, int k) {
   for (int i = pos[k] + 1; i <= n; ++i) {
      if (GoodPlace(k, i))
         return i;
   } // for
   return 0;
} // Find

void Queens(int n, int numsol = 1)  { // num of solution
   int maxsum = 0;
   int sol = 0; // solution
   memset(pos, 0, sizeof(pos));
   int k = 1;
   while (1) {
      if (k > n) { // Find one solution
         if (numsol > 1) { // all solutions
            ++sol;
            cout << "\n solution " << sol << ". ";
            Print(pos, 1, n);
            int s = SumVal(pos,n);
            cout << " Sum val = " << s;
            if (maxsum <= s) {
               maxsum = s;
               memcpy(posmax, pos, sizeof(pos));
            }
            k = n;
            continue; // while
         } // if numsol
         else { // one solution
            Print(pos, 1, n);
            return;
         }
      } // k > n
      if (k < 1) {
         if (numsol > 1) { // all solutions
```

```
            cout << "\n Total " << sol << " solution(s).";
            cout << "\n max sum = " << maxsum;
            Print(posmax, 1, n, "\n max solution: ");
            return;
          }
          else { // one solution
            cout << "\n No solution";
            return;
          }
        } // k < 1

        pos[k] = Find(n, k);

        if (pos[k] > 0) ++k;
        else --k;

      } // while
    } // Queens

    void Run() {
      srand(time(NULL));
      int n = 8;
      Gen(n);
      Print(a, 1, n, "\n a: " );
      Queens(n,2) ;
    } // Run
    main() {
      Run();

      //-----------------------------------------
      cout << endl << " T H E   E N D . ";
      return 0;
    }
```

Word Counting

Cho luồng vào gồm không quá 106 từ, mỗi từ chỉ bao gồm gồm các chữ cái viết hoa không dấu và có độ dài không quá 20 ký tự. Các từ phân cách bởi các khoảng trống hoặc dấu xuống dòng. Input file được kết thúc bằng dấu châm. Hãy đếm xem có bao nhiêu từ khác nhau, mỗi từ xuất hiện bao nhiêu lần. Kết quả được liệt kê theo thứ tự từ điển.

Ví dụ

File WORD.INP

```
BEGIN FOR  BEGIN FOR
BEGIN END END
PROCEDURE BEGIN WHILE DO BEGIN END
WHILE DO BEGIN END
DO WHILE REPEAT UNTIL CASE END
CASE END UNTIL CASE END
BEGIN END FUNCTION BEGIN END
END PROCEDURE
FOR PROCEDURE WHILE
TO PROCEDURE PROCEDURE
DO PROCEDURE FUNCTION
REPEAT PROCEDURE
.
```

Kết quả

```
11
BEGIN 8
CASE 3
DO 4
END 10
FOR 3
FUNCTION 2
PROCEDURE 7
REPEAT 2
TO 1
UNTIL 2
WHILE 4
```

Algorithm

Binary Search Tree

```
/*****************************************
  WORD.CPP (Word Counting)

  23/08/20 10:10

*****************************************/
#include <iostream>
#include <fstream>
#include <windows.h>
#include <string.h>

using namespace std;

const char * fn = "WORD.INP";
```

```
  typedef struct node{
            string V; // value
            int C; // count
            node * L;
            node * R;
          } node ;

  node * t; // tree
  string word[200];
  int wlen;

  node *NewNode(string s, int count = 0, node * l = NULL, node *r =
NULL) {
     node* e = new node;
     e->V = s;
     e-> C = count;
     e->L = l;
     e->R = r;
     return e;
  } // NewNode

  int Cmp(const string & x, const string &y) {
     int lenx = x.length();
     int leny = y.length();
     int len = min(lenx, leny);
     for (int i = 0; i < len; ++i) {
       if (x[i] != y[i])
         return (x[i] < y[i]) ? -1 : 1;
     } // for
     if (lenx == leny) return 0;
     return (lenx < leny) ? -1 : 1;
  } // Cmp

  // number of nodes
  int Card(node * t) {
     if (t == NULL) return 0;
     return Card(t->L)+ Card(t->R) + 1;
  } // Card

  void Search(string s) {
     if (t == NULL) {
       t = NewNode(s,1);
       return;
     } //  if
     node * p = t;
     while (1) {
       int d = Cmp(s, p->V);
       if (d == 0) {
         ++p->C;
         return;
       }
       if (d < 0) { // s < V: must to left
         if (p->L != NULL) p = p->L; // to left
         else {
           p->L = NewNode(s,1);
           return;
         }
       }
       else { // s > V; must to right
         if (p->R != NULL) p = p->R; // to right
```

```
        else {
          p->R = NewNode(s,1);
          return;
        }
      }
    } // while
  } // Search

  void PrintLNR(node *t) { // LNR
    if (t == NULL) return;
    PrintLNR(t->L);
    cout << "\n " << t->V << " " << t->C;
    PrintLNR(t->R);
  } // PrintLNR

  void Sort(int d, int c) {
    int i = d, j = c;
    string m = word[(d+c)/2]; // midle
    string t;
    while (i <= j) {
      while  (Cmp(word[i],m) < 0) ++i;
      while  (Cmp(word[j],m) > 0) --j;
      if (i <= j) {
        t = word[i]; word[i] = word[j]; word[j] = t;
        ++i; --j;
      }
    } // while
    if (d < j) Sort(d,j);
    if (i < c) Sort(i,c);
  } // Sort

  void PrintCount() {
    int n = 0;
    int k = 0;
    for (int i = 0; i < wlen; ++i) {
      if (word[i] != word[k]) {
        ++n;
        cout << "\n " << word[k] << ": " << i-k;
        k = i;
      }
    } // for
    if (wlen-k > 0) {
      ++n;
      cout << "\n " << word[k] << ": " << wlen-k;
    }
    cout << "\n Total " << n << " word(s).";

  } // PrintCount

  void Run() {
    string s;
    ifstream f(fn);
    t = NULL;
    wlen = 0;
    while (1) {
      f >> s;
      if (s == ".") break;
      word[wlen++] = s;
      Search(s);
      // cout << "\n " << s;
    }
```

```
    f.close();
    // Final
    cout << Card(t);
    PrintLNR(t);
    Sort(0, wlen-1);
    cout << "\n --------------------";
    PrintCount();
  } // Run

    main() {
    Run();

    //-----------------------------------------------
    cout << endl << " T H E   E N D . ";
    return 0;
  }
```

Pack and Say

Cho dãy s gồm các chữ số viết liền nhau hãy viết lại dãy số này dưới dạng các cặp số viết liền nhau XY với ý nghĩa là có tối đa X chữ số Y đứng cạnh nhau trong dãy s.

Ví dụ

s = 777777111111111111100033333

sẽ được viết thành 671313053

```
/*****************************************
   PS.CPP (Pack and Say)
   22/08/20 19:07

*****************************************/
#include <iostream>
#include <fstream>

using namespace std;

const char * fn = "PS.INP";

string ToStr(int n) {
   string s = "";
   while (n != 0) {
      s = (char)((n % 10)+'0') + s;
      n /= 10;
   } // while
   return s;
} // ToStr

void Run() {
   string inp;
   string out;
   ifstream f(fn);
   int sotest;
   f >> sotest;
   for (int i = 1; i <= sotest; ++i) {
      f >> inp;
         cout << "\n Test no " << i << ". " << inp;
         out = "";
         int k = 0;
         int len = inp.length();
         for (int j = 0; j < len; ++j) {
           if (inp[j] != inp[k]) {
               //cout << "\n " << inp[k] << ":" << j-k;
              out += ToStr(j-k) + inp[k];
           k = j;
           } // if
         } // for j
         // the rest
         if ((len-k) > 0)
              out += ToStr(len-k) + inp[k];
         cout << " -> " << out;
   } // for i
   f.close();
```

```
} // Run

main() {
   Run();
   //-----------------------------------------------
   cout << endl << " T H E   E N D . ";
   return 0;
}
```

Sequences

Cho số nguyên dương N. Hãy cho biết có bao nhiêu dãy số nguyên dương có tổng các phần tử trong dãy bằng N.

Dữ liệu vào: dòng đầu tiên chứa số nguyên T là số test, mỗi test ghi một số nguyên dương N. $1 \le T \le 20$, $1 \le N \le 1018$.

Kết quả: Mỗi test ghi ra một số nguyên duy nhất là số dư của kết quả tìm được khi chia cho 123456789.

Ví dụ

N = 3

Kết quả

4

Bốn dãy kết quả như sau:

1, 1, 1
1, 2
2, 1
3

Algorithm

Công thức

$S(N) = 2^{N-1}$

$$S(n) = 2^{n-1}$$

Gọi S(n) là hàm cho số phương án viết n thành tổng các dãy số.

Chia các phương án thành n lớp không giao nhau: lớp thứ i có số đứng đầu là i.

Ví dụ, n = 4 gồm các lớp sau: (thay vì viết 1+1+1+1 ta viết gọn: 1111)

1 đứng đầu: 1111, 112, 121, 13
2 đứng đầu: 211, 22
3 đứng đầu: 31
4 đứng đầu: 4

Nếu i đứng đầu thì còn lại n-i và số phương án của lớp này sẽ là S(n-i).

Tổng hợp lại ta có S(n) = S(n-1) + S(n-2) +…+ S(n-n)

Ta chứng minh quy nạp rằng $S(n) = 2^{n-1}$.

Base: $n = 1$: $S(1) = 1$ vì chỉ có duy nhất số 1.

Gỉa thiết quy nạp: Giả sử $S(n) = 2^{n-1}$

Ta cần chứng minh $S(n+1) = 2^n$.

Ta có

$S(n+1) = S(n+1-1) + S(n+1-2) +…+ S(n+1-(n+1)) =$

$= S(n) + (S(n-1) + \dots + S(n - n)) = S(n) + S(n) = 2S(n) = 2.2^{n-1} = 2^n.$

```
/*****************************************
   SEG.CPP (Sequences)
   23/08/20 10:10
*****************************************/
#include <iostream>
#include <fstream>

using namespace std;

typedef long long Long;
const char * fn = "SEQ.INP";

const Long M = 123456789;

// Tinh nhanh z = a^b % m
Long Exp(Long a, Long b, Long m) {
   Long z = 1;
   while (b != 0) {
      if (b & 1) { // Odd b: b % 2 = 1
         z = z*a % m;
      }
      a = (a * a) % m;
      b >>= 1; // b = b / 2
   }
   return z;
} // Exp

void Run() {
   ifstream f(fn);
   int sotest;
   f >> sotest;
   int n, m = M;
   for (int i = 1; i <= sotest; ++i) {
      cout << "\n Test no " << i << ". ";
      f >> n;
      cout << " n = " << n << " : " << Exp(2,n-1,m);
   } // for
   f.close();
} // Run

  main() {
   Run();
   //-----------------------------------------
   cout << endl << " T H E   E N D . ";
   return 0;
}
```

SEQ.INP
5
3
5
10
20
12345678912345678

Palindrome

IOI Pekin 2000

Cho string s dài tối đa 2000 ký tự chứa các chữ cái. s được gọi là *palindrome* nếu các phần tử các đều hai đầu giông nhau. Hãy cho biết cần xóa khỏi s tối thiểu k phần tử để string còn lại là một palindrome (sau khi xóa thì các phần tử tự động sát lại với nhau).

Thuật toán Dynamic Programming

Sơ đồ

Bước 1. Lập hệ thức quy hoạch động: Viết hàm biểu diễn lời giải tại bước thứ i phụ thuộc vào các bước trước hoặc sau bước i.

Bước 2. Lập trình cho phương án 1 để test với dữ liệu nhỏ.

Bước 3. Cải tiến: Dùng mảng 2 chiều để cài đặt các bước.

Bước 4. Cải tiến tiếp: Dùng mảng1 chiều hoặc vài biến phụ để cài đặt.

Bài giải

Gọi s[0..n-1] là input string, n = len(s). Gọi D(i,j) là chiều dài lớn nhất của palindrome nhận được khi giải bài toán với s[i:j].

Ví dụ

s[0:8] = abcadcbbc										
i	0	1	2	3	4	5	6	7	8	D[2,3] = 1, D[2,2] = 1, D[0,3] = 3,
s	a	b	c	a	d	c	b	b	c	D[6,7] = 2, D[4,8] = 4, D[2,5] = 3

Vài tính chất của hàm D:

1. D là hàm 2 ngôi

2. D(i,i) = 1, 0 ≤ i < n (*chứa đúng 1 kí tự*)

3. D(i, j) = 0 nếu i > j (*quy ước*)

4. D(i,j) = D(i+1, j-1) + 2, nếu s[i] = s[j]

   = max {D(i, j-1), D(i+1,j)}, nếu s[i] ≠ s[j]

Đáp số của bài khi đó sẽ là n – D(0,n-1)

```
/*****************************************
   PAL.CPP (Palindrome)
   23/08/20 10:10
*****************************************/
#include <iostream>
#include <windows.h>

using namespace std;
string s;
const int MN = 2001;
int d[MN][MN];
// int a[MN], b[MN];
```

```
void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
} // Go

// 2 array 1D
int Ver3() {
    int n = s.length();
   int *a = new int[n+1];
   int *b = new int[n+1];
   int *c;
   for (int i = 1; i < n; ++i) a[i] = 0;
   a[n-1] = 1;
   for (int i = n-2; i >= 0; --i) {
       b[i] = 1;
     for (int j = i+1; j < n; ++j) {
       b[j] = (s[i] == s[j]) ? a[j-1] + 2
              : max(b[j-1], a[j]);
     } // for j
     //memcpy(a, b, sizeof(a));
     c = a; a = b; b = c;
   } // for i
    return a[n-1];
} // Ver3


// using 2D array d
int Ver2() {
   memset(d, 0, sizeof(d));
   int n = s.length();
   for (int i = 0; i < n; ++i) {
     d[i][i] = 1;
   }
   for (int i = n-2; i >= 0; --i) {
     for (int j = i+1; j < n; ++j) {
       d[i][j] = (s[i] == s[j]) ? d[i+1][j-1] + 2
              : max(d[i][j-1], d[i+1][j]);
     } // for j
   } // for i
    return d[0][n-1];
} // Ver2

// Recursive
int D(int i, int j) {
   if (i > j) return 0;
   if (i == j) return 1;
   return (s[i] == s[j]) ? D(i+1, j-1)+ 2
          : max(D(i, j-1), D(i+1, j));
} // D

void Run() {
   s = "axbcdefghiklmnnmlkihgfedycbaaxbba";
   int n = s.length();
   cout << "\n input: " << s;
   //cout << "\n " << n - D(0,n-1);
   //cout << "\n " << n - Ver2();
   cout << "\n " << n  - Ver3();
} // Run

  main() {
```

```
    Run();
    //-----------------------------------------------
    cout << endl << " T H E   E N D . ";
    return 0;
}
```

Awards

Cần chia hết m phần thưởng cho n học sinh theo một danh sách xếp từ giỏi trở xuống. Em đứng trên nhận số phần thưởng không ít hơn em đứng dưới trong danh sách. Hỏi có bao nhiêu cách chia thưởng.

Similar Problems

Frog

Một robot ếch có thể nhảy k bước theo đường thẳng với độ dài khác nhau b(1), b(2), …, b(k).

Cho biết có bao nhiêu cách khác nhau để ếch nhảy theo đường thẳng dài N đơn vị.

Ví dụ

N = 3, k = 2, b[1] = 1, b[2] = 2.

Kết quả

3

Ba cách nhảy như sau:

1, 1, 1
1, 2
2, 1

Trả tiền

Có k loại tiền khác nhau mệnh giá b(1), b(2), …, b(k).

Cho biết có bao nhiêu cách tạo thành tập tiền có tổng N đồng.

Ví dụ

N = 4, k = 5,
b[1] = 1, b[2] = 2, b[3] = 5, b[4] = 10, b[5] = 20.

Kết quả

3
Ba cách tạo tổng như sau:
1, 1, 1, 1
1, 1, 2
2, 2

Aug. 22 2020