# Lý thuyết số

## Số nguyên tố

### Sàng số nguyên tố

|  |
| --- |
| //SangNT  #include<bits/stdc++.h>  using namespace std;  vector<int> sang\_nt(){  vector<int>v(1000001, 1);  v[0] = 0;  v[1] = 0;  for(int i = 2; i <= sqrt(1000000); i++){  if(v[i] == 1){  for(int j = i\*i; j <= 1000000; j+=i){  v[j] = 0;  }  }  }    return v;  }  int main(){  vector <int>v = sang\_nt();  for(int i = 1; i <= 100; i++){  if(v[i] == 1){  cout << i << " ";  }  }  } |

### Sàng số nguyên tố trên đoạn

|  |
| --- |
| //  #include <bits/stdc++.h>  using namespace std;  vector <int> sang\_doan(int l, int r){  vector <int> v(r-l+1, 1);  for(int i = 2; i <= sqrt(r); i++){  {  for(int j = max(i\*i, (l+i-1)/i\*i); j<= r;j+=i){  v[j-l] = 0;  }  }  }  return v;  }  int main(){  int l, r;  cin >> l >> r;  vector <int> v = sang\_doan(l, r);  for(int i = max(2, l); i <= r; i++){  if(v[i-l] == 1){  cout << i << " ";  }  }  } |

* Ý tưởng: lưu trữ các giá trị kiểm tra nguyên tố vào 1 mảng có (l-r+1) phần tử
* Duyệt qua các bội của I (từ l đến sqrt(r)) và đánh dấu nó bằng 0

### Ước số nguyên tố nhỏ nhất

|  |
| --- |
| //sàng biến đổi  vector<int> sang\_bd(){  vector<int>v(100001);  for(int i = 0; i <= 100000; i++)  v[i] = i;    for(int i = 2; i <= sqrt(100000); i++){  if(v[i] == i){  for(int j = i\*i; j <= 100000; j+=i){  if(v[j] == j){  v[j] = i;  }  }  }  }    return v;  } |

### N! chia hết cho px

***Ý tưởng***

* Đếm xem từ 1 đến n có bao nhiêu số chia hết cho p. Chú ý trường hợp 1 số nào tích của nhiều số p

|  |
| --- |
| //Cách 1  int dem(int n, int p){  int cnt = 0;  for(int i = 1; i <= n; i++){  if(i % p == 0){  int tmp = i;  while(tmp % p == 0){  cnt++;  tmp /= p;  }  }  }  return cnt;  } |

***Cách 2(Legendre)***

Sử dụng công thức

|  |
| --- |
| (vòng lặp khi mẫu số nhỏ hơn tử số) |

|  |
| --- |
| //  int dem(int n, int p){  int cnt = 0;  for(int i = p; i <= n; i\*=p)  cnt += n/i;  return cnt;  } |

### Số chữ số 0

Đếm số lượng chữ số 0 của n!

***Ý tưởng***

* Số 0 được tạo thánh từ tích với số 10. 10 được phân tích = 2x5. Mà số lần xuất hiện của 5 ít hơn 2 => Cần tìm số lần mà 5 được phân tích trong n!
* Giống bài trên

### Số hoàn hảo

Nếu p là số nguyên tố và cũng là số nguyên tố thì (-1) là số hoàn hảo

|  |
| --- |
| //  int ktra\_nt(int n){  if(n < 2)  return 0;  for(int i = 2; i <= sqrt(n); i++)  if(n % i == 0)  return 0;  return 1;  }  vector <long long> init\_hh(){  vector <long long>v;  for(int i = 2; i <= 32; i++){  if(ktra\_nt(i) && ktra\_nt(pow(2, i)-1))  v.push\_back( pow(2, i-1)\*(pow(2, i)-1) );  }  return v;  } |

### Phi hàm Ơ-le

* Đếm số lượng số nguyên tố cùng nhau với n

|  |
| --- |
| //phi bằng tích n x (1-1/p)  // với p là các số nguyên tố và n chia hết cho p  //n(1-1/p) có thể biến đổi = n – n/p để tránh sai số |

|  |
| --- |
| //Phi ham Ole  long long phi(long long n){  long long res = n;  for(int i = 2; i <= sqrt(n); i++){  if(n % i == 0){  res = res - res/i;  while(n % i == 0)  n /= i;  }  }  if(n != 1){  res = res - res/n;  }  return res;  } |

|  |
| --- |
| //Sang phi ham ole  vector<int> sang\_ole(){  vector<int>v(1000001);  for(int i = 1; i <= 1000000; i++)  v[i] = i;    //do so nguyen to co phi ham ole = chinh no - 1  for(int i = 2; i <= 1000000; i++){  if(v[i] == i){  v[i] -=1;  //duyet cac boi cua i  for(int j = i \* 2; j <= 1000000; j+=i)  v[j] = v[j] - v[j]/i;  }  }    return v;  } |

### Số ước của n!

|  |
| --- |
| Giả sử n! có thể chia hết cho tối đa px. Giả sử có nhiều hơn 1 p. Ta có thể viết như sau  Khi đó số lượng ước của n! = ) ) )… )  với p là các số nguyên tố |

|  |
| --- |
| //  //tim bac cua p trong n!  int tim\_bac(int p, int n){  int res = 0;  for(int i = p; i <= n; i\*=p)  res += n/i;  return res;  }  int ktra\_nt(int n){  if(n < 2)  return 0;  for(int i = 2; i <= sqrt(n); i++)  if(n % i == 0)  return 0;  return 1;  }  long long dem\_uoc(int n){  long long res = 1;  for(int i = 2; i <= n; i++){  if(ktra\_nt(i)){  res \*= (tim\_bac(i, n) + 1);  }  }  return res;  } |

### lcm(sum)

Cho số n. Yêu cầu tính tổng lcm(1, n) + lcm(2, n) + lcm(3, n) + … + lcm(n, n)

Với lcm(a, b) là bcnn(a, b)

|  |
| --- |
| Sử dụng công thức  Với k là ước của n  Ví dụ với lcm(i, 6). Với i = 1, 2…6 sẽ bằng )\*6/2 |

|  |
| --- |
| //  vector <int> phi\_ham\_ole(){  vector<int> v(1000001);  for(int i = 1; i <= 1000000; i++)  v[i] = i;    for(int i = 2; i <= 1000000; i++){  if(v[i] == i){  v[i]--;  for(int j = i \* 2; j <= 1000000; j+=i){  v[j] = v[j] - v[j]/i;  }  }  }  return v;  }  vector<long long>sang\_tong(){  vector <int> ol = phi\_ham\_ole();  vector <long long> v(1000001, 0);  for(int i = 1; i<= 1000000; i++){  for(int j = i ; j <= 1000000; j+=i){  v[j] += i \* ol[i];  }  }  return v;  } |

### x^y > y^x

* Có 2 trường hợp có thể xảy ra

x <= 3

* x = 0, không thể tạo x^y > y^x
* x = 1, chỉ có thể tạo với y = 1
* x = 2, có thể tạo với y = 0, y = 1, y > x (trừ trường hợp y= 3 và y = 4)
* có thể viết = f(0) + f(1) + f(y > x) – f(3) – f(4)
* x = 3, có thể tạo với y = 0, y = 1, y = 2 và y > x
* có thể viết = f(0) + f(1) + f(y > x) + f(2)

x > 3

* Có thể tạo cặp với mọi x < y

|  |
| --- |
| //  #include <bits/stdc++.h>  using namespace std;  int cnt[5];  int tim\_lon\_dau\_tien(int a[], int l, int r, int x){  int res = -1;  while(l <= r){  int m = (l + r)/2;  if(a[m] > x){  res = m;  r = m - 1;  }else  l = m + 1;  }  return res;  }  int dem(int b[], int m, int x){  int res = 0;    if(x == 0)  return 0;  else if(x == 1)  return cnt[0];  else{  res = cnt[0] + cnt[1];    int vi\_tri = tim\_lon\_dau\_tien(b, 0, m - 1, x);  if(vi\_tri != -1)  res += vi\_tri;    if(x == 2)  res = res - cnt [3] - cnt[4];  if(x == 3)  res = res + cnt [2];  }  return res;  }  int main(){  int t;  cin >> t;  while(t--){  for(int i = 0; i < 5; i++)  cnt[i] = 0;      int n, m;  cin >> n >> m;  int a[n], b[m];  for(int i = 0; i < n; i++)  cin >> a[i];  for(int i = 0; i < m; i++){  cin >> b[i];  if(b[i] <= 4)  cnt[b[i]]++;  }  sort(b, b + m);  int res = 0;  for(int i = 0; i < n; i++){  res += dem(b, m, a[i]);  }    cout << res << endl;  }  } |

## Đồng dư

1 số công thức cần ghi nhớ

(a+b)%c = [(a%c)+(b%c)]%c

(a-b)%c = [(a%c)-(b%c)]%c

(a\*b)%c = [(a%c)\*(b%c)]%c

### Tính tổng modulo

Cho 2 số nguyên n và k. Hãy tính tổng S = n%1 + n%2 + n%3 +… +n%k

|  |
| --- |
| Các số hạng của S sẽ tuần hoàn như sau  1 2 3 … 0 1 2 3 …0 1,2 3 n%k  Các số hạng 0 tại các vị trí là bội của k. Cụm cuối cùng có thể không phủ hết 1 lần k, nên tổng quát là n%k  Các cụm đầy đủ sẽ là dãy tự nhiên từ 1 đến k – 1   * Công thức tổng quát |

### Số nguyên dương thứ k không chia hết cho n

Tìm số nguyên dương thứ k không chia hết cho n

|  |
| --- |
| Giả sử có n = 5, k = 8  1 2 3 4 [5] 6 7 8 9 [10]  Nhận thấy cứ n số thì có n-1 số không chia hết cho n  lấy có thể biết được số cụm. nếu có phần dư, thì số cần tìm sẽ ở cụm tiếp theo. Nếu không, số cần tìm sẽ ở cụm hiện tại. r = k%(n-1)  Với phần dư (r == 0, số cụm = x) => số cần tìm = n \* x – 1;  Với phần dư (r != 0, số cụm = x) => số cần tìm = n \* x + r; |

### Chữ số tận cùng 1378^n

|  |
| --- |
| Nhận thấy số tận cùng của 2^n sẽ tuần hoàn theo chu kì  2^1 = 2  2^2 = 4  2^3 = 8  2^4 = 16  Số tận cùng lặp theo chu kỳ 2/4/8/6  -  1378^n, số tận cùng phụ thuộc vào 8^n  8^1 = 8  8^2 = 4  8^3 = 2  8^4 = 6 |

### (1^n+2^n+3^n+4^n+5^n)mod 5

Nhận thấy 1^n, 2^n, 3^n, 4^n đều là những số có chữ số cuối tuần hoàn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1^n | 1 | 1 | 1 | 1 |
| 2^n | 2 | 4 | 8 | 6 |
| 3^n | 3 | 9 | 7 | 1 |
| 4^n | 4 | 6 | 4 | 6 |
| Sum | 0 | 0 | 0 | 4 |

Do đó, (1^n+2^n+3^n+4^n) mod 5 dư 4 nếu n chia hết cho 4. Và (1^n+2^n+3^n+4^n) mod 5 = 0, trong các trường hợp còn lại

### Last digit a^n

Tìm chữ số tận cùng của a^n, với a và n rất lớn (1<a.n<1010\_000\_000)

Quy luật của các chữ số cuối

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 0^n | 0 | 0 | 0 | 0 |
| 1^n | 1 | 1 | 1 | 1 |
| 2^n | 2 | 4 | 8 | 6 |
| 3^n | 3 | 9 | 7 | 1 |
| 4^n | 4 | 6 | 4 | 6 |
| 5^n | 5 | 5 | 5 | 5 |
| 6^n | 6 | 6 | 6 | 6 |
| 7^n | 7 | 9 | 3 | 1 |
| 8^n | 8 | 4 | 2 | 6 |
| 9^n | 9 | 1 | 9 | 1 |

|  |
| --- |
| //  int tim\_so(string a, string n){  int x = a[a.size()-1] - '0';  if(x == 0||x == 1||x == 5||x == 6)  return x;  if(n.size() == 1)  n = "0" + n;  int y = (n[n.size()-2] - '0') \* 10;  y += n[n.size()-1] - '0';    int r2[4] = {6, 2, 4, 8};  int r3[4] = {1, 3, 9, 7};  int r4[4] = {6, 4, 6, 4};  int r7[4] = {1, 7, 9, 3};  int r8[4] = {6, 8, 4, 2};  int r9[4] = {1, 9, 1, 9};    y %= 4;  if(x == 2)  return r2[y];  if(x == 3)  return r3[y];  if(x == 4)  return r4[y];  if(x == 7)  return r7[y];  if(x == 8)  return r8[y];  if(x == 9)  return r9[y];  } |

## No Title

#### Đếm số chia hết

Một số nguyên không âm được gọi là đẹp nếu như chia hết cho một trong ba số sau: 4, 7 và 11. Hãy đếm số lượng số đẹp thỏa mãn L <= x <= R

***Gợi ý***

1. Đếm số lượng số chia hết cho 1 số nguyên x. Sử dụng công thức

* Công thức tính cận dưới (l)
* Công thức tính cận trên (r)
  + Trong đó, cận dưới là số nhỏ nhất lớn hơn hoặc bằng l chia hết cho x
  + Trong đó, cận trên là số lớn nhất nhỏ hơn hoặc bằng r chia hết cho x

2. Vì 4, 7, 11 là 3 số nguyên tố cùng nhau. Gọi số lượng số chia hết cho 1 số x là N(x). Công thức cuối cùng như sau

N = N(4) + N(7) + N(11) + N(4\*7\*11)-N(4\*7)-N(4\*11)-N(7\*11)

* Do, số lượng bội của 4, 7, 11 khi đếm sẽ có những số là bội chung nên phải trừ đi số lượng trùng này

#### Giải hệ phương trình bậc nhất

Viết chương trình tìm nghiệm tổng quát của hệ

***Gợi ý***

* Sử dụng công thức nghiệm với phương pháp Cramer. Lưu ý, phương pháp Cramer dùng để lấy nghiệm của hệ có số ẩn bằng số phương trình, chỉ áp dụng khi hệ có 1 nghiệm duy nhất.
* Giả sử với hệ tổng quát

Có thể viết hệ dưới dạng tổng quát

Ax=b

* Giả sử A có định thức khác 0. Công thức tính nghiệm của hệ

***Xét bài toán bậc 2***

* Biện luận nghiệm như sau

***Xét bài toán ban đầu***

Det(A) = ae-bd

Det(x) = ce-bf

Det(y) = af-dc

Nghiệm duy nhất

Điều kiện: Det(A)!=0

x = Det(x)/Det(A)

y = Det(y)/Det(A)

Vô nghiệm

Det(A) = 0

Vô số nghiệm

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int main(){  long long a, b, c;  long long d, e, f;  cin >> a >> b >> c >> d >> e >> f;  double delta = a\*e-b\*d;  double delta\_x = c\*e-b\*f;  double delta\_y = a\*f-c\*d;  if(delta == 0){  if( delta\_x == 0 && delta\_y == 0){  cout <<"VOSONGHIEM";  }  else  cout <<"VONGHIEM";  }  else{  double n1 = delta\_x/delta;  double n2 = delta\_y/delta;      if(abs(n1-0) <= 0.00)  n1 = 0;  if(abs(n2-0) <= 0.00)  n2 = 0;  printf("%.2lf %.2lf", n1, n2);  }  } |

#### Căn bậc 2 lồng nhau

Cho số nguyên dương n, tính

***Gợi ý***

* Có thể tính từ cuối về

|  |
| --- |
| //  #include <bits/stdc++.h>  using namespace std;  int main() {  int n;  cin >> n;    double res = 0;  for(int i = n; i >= 1; i--){  res = sqrt(i + res);  }    cout << res;  } |

## Nhị phân

### Lũy thừa

* Ý tưởng

Lũy thừa an có thể được tính nhanh với công thức như sau

* Cài dặt

|  |
| --- |
| //Vòng lặp  long long tinh\_luy\_thua(int a, int b){  int res = 1;  while(b != 0){  if(b%2 == 1){  res \*= a;  }  b/=2;  a\*=a;  }  return res;  } |

|  |
| --- |
| //Đệ quy  long long tinh\_luy\_thua(int a, int b){  if(b == 0)  return 1;  else{  long long x = tinh\_luy\_thua(a, b/2);  if(b % 2 == 1)  return x \* x \*a;  else  return x \* x;  }  } |

# Cấu trúc dữ liệu

## Vector

### Lý thuyết

* Vector là một cấu trúc dữ liệu không cần khai báo trước số phần tử (mảng động)
* Vector là một lớp với các phương thức
* Một số phương thức
* Cú pháp khai báo

|  |
| --- |
| vector<int>*v*(*size*, *value*);  //v là tên vector  //size là kích thước  // value là giá trị cho tất cả |

### push\_back()

* thêm một phần tử vào cuối vector

### size()

* Lấy ra kích thước

### max\_element(it1, it2)

* Tìm về iterator lớn nhất mảng
* Giải tham chiếu để lấy giá trị

\*max\_element(it1, it2);

### min\_element(it1, it2)

### accumulate(it1, it2, sum)

* Tính tổng từ [it1, it2) với giá trị khởi tạo là sum

### earse()

## String

### Nhập/xuất

* Lệnh nhập
  + Nếu trước khi nhập string, có phím enter, cần xử lý trôi lệnh

|  |
| --- |
| cin.ignore()  //mặc định xóa 1 ký tự trước đó |

* + getline(cin, s); //lệnh nhập

### stringstream

* Dùng để tách chuỗi. Mặc định tách tại các khoảng trắng

|  |
| --- |
| //  #include <iostream>  #include <string>  #include<sstream>  using namespace std;  int main(){  string s;  getline(cin , s);  stringstream ss(s);  string tmp;  while(ss >> tmp){  cout << tmp << " ";  }  } |

//Tách từ tại dấu phẩy hoặc các ký tự khác

|  |
| --- |
| //  //Dùng ss.good() và getline() cho token khác ví dụ trên  #include <bits/stdc++.h>  using namespace std;  int main(){  string s;  getline(cin, s);  stringstream ss(s);  string tk;  while(ss.good()){  getline(ss, tk, ',');  cout << tk << " ";  }  }  /\*  Lap, Trinh, Thi, Dau,  \*/ |

***Truyền tham chiếu***

* Ở các hàm không trả về (void), nếu truyền string là tham số, thì sau hàm đó string sẽ không bị thay đổi. Trong C++, để thay đổi string này, có thể truyền theo kiểu tham chiếu (đặt ký pháp & phía trước biến string)

|  |
| --- |
| void thay\_doi(string &s){  s[0] = ‘a’;  } |

### Pair

Khai báo

|  |
| --- |
| pair<kdl\_1, kdl\_2>  //pair lưu được 1 cặp dữ liệu ứng với 2 thuộc tính: first, second |

### Cộng hai số nguyên lớn

|  |
| --- |
| //  string cong\_2\_so(string a, string b){  while(a.size() < b.size()) a = "0" + a;  while(b.size() < a.size()) b = "0" + b;  string res = "";    int nho = 0;  for(int i = a.size() - 1; i >=0; i--){  int x = int(a[i] - '0') + int(b[i] - '0') + nho;  res = char(x % 10 + '0') + res;  nho = x/10;  }  if(nho > 0)  res = char(nho + '0') + res;    return res;  } |

### Trừ hai số nguyên lớn

|  |
| --- |
| //  string tru(string a, string b){  string res = "";  while(a.size() < b.size()) a = "0" + a;  while(b.size() < a.size()) b = "0" + b;    //dat a la so lon hon  int am = 0;  if(a < b){  swap(a, b);  am = 1;  }    int nho = 0;  for(int i = a.size() - 1; i >= 0; i--){  int x = int(a[i] - '0') - (b[i] - '0') - nho;  if(x < 0){  x += 10;  nho = 1;  }else  nho = 0;  res = char(x + '0') + res;  }  string kq = "";  int i;  for(i = 0; i < res.size(); i++)  if(res[i] != '0')  break;    for(int j = i; j < res.size(); j++)  kq += res[j];    if(am == 1)  kq = "-" + kq;  return kq;  } |

### Thuật toán boyer-moore

***Luật ký tự tồi***

* Ví dụ, xét 2 xâu T = FINDIFAHAYXECAKNEXDLE

Và xâu P = NEXDLE

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| F | I | N | D | I | F | A | H | A | Y | X | E | C | A | K | N | E | X | D | L | E |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | E | X | D | L | E |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | E | X | D | L | E |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | E | X | D | L | E |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | E | X | D | L | E |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | E | X | D | L | E |

### Bài tập

#### Chữ số La Mã

Trước khi xuất hiện chữ số Ả Rập - là các chữ số từ 0 đến 9 mà chúng ta đang sử dụng rộng rãi ngày nay - trong thời cổ đại và trung đại người ta sử dụng số La Mã. Số La Mã gồm 7 ký tự tương ứng với các số Ả Rập như sau:

|  |  |
| --- | --- |
| I | 1 |
| V | 5 |
| X | 10 |
| L | 50 |
| C | 100 |
| D | 500 |
| M | 1000 |

Và các ký tự ghép như sau:

|  |  |
| --- | --- |
| IV | 4 |
| IX | 9 |
| XL | 40 |
| XC | 90 |
| CD | 400 |
| CM | 900 |

Input

* Dòng đầu tiên là số lượng test case T (1≤T≤100).
* Mỗi test case là một số La Mã

Output

* In ra dạng thập phân của số La Mã

|  |  |
| --- | --- |
| Input | Output |
| 2  III  MMMCMXCIX | 3  3999 |

***Ý tưởng***

* Số đứng trước nhỏ hơn số đứng sau thì trừ đi số đứng trước
* Số đứng trước lớn hơn hoặc bằng số đứng sau thì cộng với số đứng sau

|  |
| --- |
| //  int vi\_tri(char c, char s[]){  int pos = 0;  for(int i = 0; i < strlen(s); i++)  if(s[i] == c){  pos = i;  break;  }    return pos;  }  int tinh\_gia\_tri(string s){  int value[] = {1, 5, 10, 50, 100, 500, 1000};  char ky\_tu[] = {'I', 'V', 'X', 'L', 'C', 'D', 'M'};    int res = value[vi\_tri(s[s.size() - 1], ky\_tu)];  for(int i = s.size() - 1; i >= 1; i--){  if(vi\_tri(s[i - 1], ky\_tu) >= vi\_tri(s[i], ky\_tu))  res += value[vi\_tri(s[i - 1], ky\_tu)];  else  res -= value[vi\_tri(s[i - 1], ky\_tu)];  }    return res;  } |

#### Số nguyên lớn chia hết cho 5

***Ý tưởng***

* Tím số tận cùng

|  |
| --- |
| //  int ktra (string s){  int res = 0, mu = 1;  for(int i= s.size() - 1; i >= 0; i--){  res += (s[i] - '0') \* mu;  mu \*= 2;  mu %= 10;  res %= 10;  }  return (res % 5 == 0);  } |

#### Số dư của 1 số nguyên lớn và long long

* Tìm số dư của phép chia một số nguyên lớn với long long

***Ý tưởng***

* Chia lấy phần dư từ trái sang, duyệt qua mỗi đơn vị lại nhân 10 rồi cộng với số dư trước đó, sau đó chia dư

Ví dụ: 1234 chia dư 5

+, Lấy 1 chia dư 5 = 1

+, sau đó lấy phần dư (1\*10)cộng với chữ số tiếp theo rồi chia dư 5 như sau: (1\*10+2)%5 = 2

+, tiếp tục (2\*10+3)%5 = 3

+, cuối cùng (3 \* 10 + 4 ) % 5 = 4;

|  |
| --- |
| //  int tim\_du(string s, long long x){  int res = 0;  for(int i = 0; i < s.size(); i++){  res = (res \* 10 + (s[i] - '0'))%x;  res % x;  }  return res;  } |

#### Tìm ước chung lớn nhất của số nguyên lớn

* Tìm ước chung lớn nhất của một số nguyên lớn và long long

***Gợi ý***

* Dựa vào công thức đệ quy. tìm ucln a, b

return ucln(b, a%b)

* Tìm số dư trước, rồi lắp vào công thức

|  |
| --- |
| //  long long tim\_du(string s, long long x){  long long res = 0;  for(int i = 0; i < s.size(); i++){  res = (res \* 10 + (s[i] - '0'))%x;  res %= x;  }  return res;  }  long long tim\_uoc(long long a, long long b){  if(b == 0)  return a;  return tim\_uoc(b, a%b);  } |

#### (a^b)modc

(Với a là số nguyên lớn)

***Gợi ý***

* Tính a mod c trước
* Sau đó tính mod với lũy thừa nhị phân

|  |
| --- |
| //  long long tim\_du(string s, long long mod){  long long res = 0;  for(int i = 0; i < s.size(); i++){  res = res \* 10 +(s[i] - '0');  res %= mod;  }  return res;  }  long long lt(long long x, long long y, long long mod){  long long res = 1;  while(y){  if(y % 2 == 1){  res \*= x;  res %= mod;  }  y /= 2;  x \*= x;  x %= mod;  }  return res;  } |

#### Chia hết cho 2^x

***Gợi ý***

* Ví dụ xâu 111000 và 3
* Xâu trên chia hết cho 3 khi 3 bit cuối cùng phải bằng 0 hết

|  |
| --- |
| //  int ktra(string s, int x){  int cnt = 0;  for(int i = s.size() - 1; i >= 0; i--){  if(cnt == x)  break;  if(s[i] == '1')  return 0;  cnt++;  }  return 1;  } |

#### Bài 7. Số lớn nhất nhỏ nhất

* Cho 1 số, bạn được đổi chỗ tối đa 1 lần, hãy đưa ra số lớn nhất nhỏ hơn chính nó

***Ý tưởng***

* Nếu dãy tăng dần, thì không có phương án nào
* Vị trí đứt đoạn đầu tiên sẽ cần thay. Số thay cho số đó là số lớn nhất nhưng nhỏ hơn số này (về bên trái nhất)

1 2 4 3 3 5 5

Trường hợp này sẽ đổi 4 cho 3 (trái)

|  |
| --- |
| //  string tim\_lon\_nhat(string s){  int i = s.length() - 2;  while(i >= 0 && s[i] <= s[i + 1]){  i --;  }  if(i == -1)  return "-1";  int j = s.length() - 2;  while(s[i] <= s[j]){  j --;  }  while(s[j] == s[j - 1])  j--;  swap(s[i], s[j]);  return s;  } |

#### Xâu con có k ký tự khác nhau

* Cho chuỗi s và xâu k. Tìm số xâu con có k ký tự khác nhau

***Gợi ý***

* Duyệt trâu

#### Tính giai thừa các chữ số

* Chú ý một số trường hợp đặc biệt

4! = 3! \* 2! \* 2!

6! = 5! \* 3!

8! = 2! \* 2! \* 2! \* 7!

9! = 3! \* 3! \* 2! \* 7!

|  |
| --- |
| //  string so\_ln(string x){  string res = "";  for(char c: x){  if(c == '4'){  res += "322";  }else if(c == '6')  res += 35;  else if(c == '8')  res += "2227";  else if(c == '9')  res += "3327";  else if(c != '1' && c != '0')  res += c;  }  sort(res.begin(), res.end(), greater<char>());  return res;  } |

#### Giải mã tăng giảm

## Set

* Là một cấu trúc dữ liệu mà các phần tử trong đó phải khác nhau đôi một
* Set là một tập hợp có thứ tự
* Phù hợp với bài toán đếm số phần tử khác nhau trong mảng

### insert ()

* thêm một phần tử vào trong set

|  |
| --- |
| //  set\_name.find(element)  //  Trả về vị trí nếu tìm thấy  Trả về set\_name.end() nếu không tìm thấy |

### find()

* O(log(n))

### size()

### earse()

## Multi-set

* Giống set, nhưng lưu được các phần tử có giá trị giống nhau
* Khi xóa (earse), thì sẽ xóa hết các phần tử

## Map

* Được cài đặt bằng bảng băm
* Phù hợp với bài toán đếm số lần xuất hiện của các phần tử

|  |
| --- |
| map<kdl1, kdl2>map\_name; |

* Map lưu các cặp pair gồm phần tử first (key) và second(value)
* Các key phải là duy nhất

### Duyệt map

* Map được tạo bởi các cặp pair, nên có thể duyệt qua foreach

|  |
| --- |
| for(pair<int, int> x: map\_name){  cout << x.first <<" "<<x.second <<endl;  } |

* Có thể dùng tử khóa auto (đại diện cho biến)

|  |
| --- |
| //  for(auto x: map\_name){  cout << x.first <<" "<<x.second <<endl;  } |

## MultiMap

* Giống map nhưng lưu được các key giống nhau
* Khi thêm, thêm kiểu pair

|  |
| --- |
| //  mp.insert{first, second} |

# Sắp xếp

### Các thuật toán sắp xếp

#### Interchange sort

|  |
| --- |
| //  void interchange\_sort(int a[], int n){  for(int i = 0; i < n - 1; i++){  for(int j = i + 1; j < n; j++){  if(a[i] > a[j]){  swap(a[i], a[j]);  }  }  }  } |

#### Selection sort

|  |
| --- |
| //  void selection\_sort(int a[], int n){  for(int i = 0; i < n - 1; i++){  int id = i;  for(int j = i + 1; j < n; j++){  if(a[j] < a[id])  id = j;  }  if(i != id)  swap(a[i], a[id]);  }  } |

#### Insertion sort

* Xét từ vị trí index = 1, tìm trong dãy trước nó, xem phần tử nào đầu tiên nhỏ hơn a[index], thì chèn a[index] ngay sau phần tử đó

|  |
| --- |
| //  void insert\_sort(int a[], int n){  for(int i = 1; i < n; i++){  //tim vi tri phu hop cho a[i]  int j = i;  int tmp = a[i];  while(j >0 && a[i] < a[j-1])  j--;  for(int k = i; k > j; k--)  a[k] = a[k - 1];  a[j] = tmp;  }  } |

#### Bubble sort

* Xét bài toán sắp xếp tăng dần
* Xét 2 phần tử cạnh nhau, phần tử lớn hơn sẽ nổi lên(lùi về sau dãy)

|  |
| --- |
| //  void buble\_sort(int a[], int n){  for(int i = 0; i < n; i++){  for(int j = 0; j < n - 1; j ++){  if(a[j] > a[j + 1])  swap(a[j], a[j + 1]);  }  }  } |

#### quick sort

* Phân hoạch Hoare (Onlogn)

***Ý tưởng***

Ví dụ, có mảng

[1, 7, 2, 3, 5, 6, 8, 0]

* Tại mỗi bước, thuật toán sẽ chia mảng thành 2 nửa, nửa bên trái nhỏ hơn hoặc bằng 1 giá trị pivot nào đó, nửa bên phải lớn hơn hoặc bằng giá trị pivot
* Cách chọn pivot: (đầu, giữa, cuối hoặc random, trung vị )
* Ví dụ này, chọn pivot ở giữa

|  |
| --- |
| //  [1, 7, 2, 3, 5, 6, 8, 0]   * Phần tử ở giữa là 3 (pivot = 3) * Đặt 2 con trỏ ở đầu và cuối mảng. Duyệt về giữa, đổi chỗi 2 phần tử vi phạm đầu tiên. Trong trường hợp này, 7 và 0 là 2 phần tử vi phạm   Dãy số mới [1, 0, 2, 3, 5, 6, 8, 7]   * Thuật toán dừng, khi 2 con trỏ bằng vị trí nhau * Gọi đệ qui 2 nửa mảng |

### Hàm sort()

sort(begin, end, comparator)

* Trong đó comparator là một hàm bool để tùy biến hàm sắp xếp

|  |
| --- |
| //  //Ví dụ, sắp dãy tăng dần  //Trả về true nếu muốn đứng trước, trả về false nếu muốn đứng sau  bool compar(int x, int y){  if(x < y)  return true;  return false;  }  int main(){  vector<int>v(5);  for(int i = 0; i < 5; i++){  cin >> v[i];  }    sort(v.begin(), v.end(), compar);  for(auto el: v)  cout << el << " ";  }  /\*  1 9 5 3 6  \*/ |

|  |
| --- |
| //cmp với string  bool cmp(string a, string b){  return a.size() < b.size();  }  int main(){  vector<string>v(5);  for(auto &el: v)  getline(cin, el);    sort(v.begin(), v.end(), cmp);  for(auto el: v)  cout << el << " ";  } |

*=>Tuy nhiên 2 từ cùng tiêu chí so sánh mà chưa được xử lý, sẽ đứng ngẫu nhiên*

### stable\_sort ()

* Sử dụng cho bài toán sắp xếp, nều cùng tiêu chí, thì giữ nguyên vị trí ban đầu

### Bài tập

#### Sắp xếp theo tần số

* Sắp xếp các phần tử mảng theo tần số xuất hiện, nếu cùng tần số, thì in ra theo thứ tự
* ***Gợi ý***
  + Viết cmp riêng cho hàm stable\_sort với map là một dữ liệu được khai báo toàn cục

|  |
| --- |
| //  #include <bits/stdc++.h>  #include <vector>  using namespace std;  map<int, int>mp;  bool cmp(int a, int b){  return mp[a] > mp[b];  }  int main(){  int n;  cin >> n;  int a[n];  for(int i = 0; i < n; i++){  cin >> a[i];  mp[a[i]]++;  }    stable\_sort(a, a + n, cmp);  for(int el: a)  cout << el << " ";    } |

#### Xếp gạch

* Một viên gạch có độ cứng k, sẽ xếp chồng được k viên gạch khác. Tính số gạch tối đa xếp chồng được

***Gợi ý***

* Luôn có viên đầu tiên được xếp
* Độ cứng được cập nhật với công thức = min(docung-1, docung[i]), với i là vị trí được xét đến (i tính từ 1 đến n-1)

|  |
| --- |
| //  int dem\_gach(int a[], int n){  int res = 1;  int do\_cung = a[0];  for(int i = 1; i < n; i++){  if(do\_cung >= 1)  res ++;  else  break;  do\_cung = min(do\_cung - 1, a[i]);  }    return res;  } |

#### Lập lịch

* Ca sĩ Lệ Rơi có n lịch diễn với khoảng thời gian bắt đầu và kết thúc cho trước. Xác định số show diễn tối đa ca sĩ Lệ Rơi có thể tham gia. (thời gian bắt dầu phải khác thời gian kết thúc trước đó)

***Gợi ý***

* Tham lam theo thời gian bắt đầu
* B1: Sắp xếp tăng dần theo thời gian kết thúc
* B2: Nếu thời gian bắt đầu công đoạn sau lớn hơn thời gian bắt đầu công đoạn trước thì lấy (luôn chọn công đoạn 1), chú ý cập nhật mốc kết thúc

|  |
| --- |
| //  bool cmp(pair<int, int>p1, pair<int, int>p2){  return p1.second < p2.second;  }  int dem\_show(pair<int, int>p[], int n){  int res = 1;  int kt = p[0].second;  for(int i = 1; i < n; i++){  if(p[i].first > kt){  res ++;  kt = p[i].second;  }  }    return res;  } |

#### Hai phần tử có tổng bằng k

* Cho mảng với các phần tử cho trước, đếm số cặp có tổng bằng k

***Gợi ý***

* B1: Sắp xếp
* B2: Tại mỗi phần tử, tìm nhị phân xem có bao nhiều phần tử bằng hiệu của k với chính nó (vị trí tìm tính từ vị trí tiếp theo đến hết dãy)

|  |
| --- |
| //  //nlogn  //lien tuc ve trai  int tim\_dau\_tien(int a[], int l, int r, int x){  int res = -1;  while(l <= r){  int m = (l + r)/2;  if(a[m] == x){  res = m;  r = m - 1;  }else if(a[m] < x){  l = m + 1;  }else  r = m - 1;  }  return res;  }  //lien tuc ve phai  int tim\_cuoi\_cung(int a[], int l, int r, int x){  int res = -1;  while(l <= r){  int m = (l + r)/2;  if(a[m] == x){  res = m;  l = m + 1;  }else if(a[m] < x){  l = m + 1;  }else  r = m - 1;  }  return res;  }  int main(){  int t;  cin >> t;  while(t--){  int n, k;  cin >> n >> k;  int a[n];    for(int i = 0; i < n; i++){  cin >> a[i];  }  sort(a, a + n);  int res = 0;  for(int i = 0; i < n - 1; i++){  int l = tim\_dau\_tien(a, i + 1, n - 1, k - a[i]);  int r = tim\_cuoi\_cung(a, i + 1, n - 1, k - a[i]);  if(l != -1)  res = res + (r - l + 1);  }    cout << res << endl;  }  } |

#### Cặp số có tổng nhỏ hơn k

***Gợi ý***

* Tại vị trí i, tìm từ đoạn [i + 1, n - 1] xem index cuối cùng có giá trị < k – a[i]
* Nếu vị trí này khác – 1, kết quả được cộng thêm với (vị trí – i)

|  |
| --- |
| //  //tim lien tuc ve phai  int last\_post(int a[], int l, int r, int x){  int res = -1;  while(l <= r){  int m = (l + r)/2;  if(a[m] < x){  res = m;  l = m + 1;  }else if(a[m] >= x)  r = m - 1;  }  return res;  }  int main(){  int t;  cin >> t;  while(t--){  int n, k;  cin >> n >> k;  int a[n];    for(int i = 0; i < n; i++){  cin >> a[i];  }  sort(a, a + n);  int res = 0;  for(int i = 0; i < n - 1; i++){  int x = last\_post(a, i + 1, n - 1, (k - a[i]) );  if(x != -1)  res += (x - i);  }    cout << res << endl;  }  } |

#### Cặp số có tổng lớn hơn k

|  |
| --- |
| //  //tim vi tri dau tien (> k - a[i])  //tim lien tuc ve trai  int pos(int a[], int l, int r, int k){  int res = -1;  while(l <= r){  int m = (l + r)/2;  if(a[m] > k){  res = m;  r = m - 1;  }else{  l = m + 1;  }  }  return res;  }  int main(){  int t;  cin >> t;  while(t--){  int n, k;  cin >> n >> k;  int a[n];    for(int i = 0; i < n; i++){  cin >> a[i];  }  sort(a, a + n);  int res = 0;  for(int i = 0; i < n - 1; i++){  int x = pos(a, i + 1, n - 1, (k - a[i]) );  if(x != -1)  res += (n - x);  }    cout << res << endl;  }  } |

#### Sắp dãy con

* Cho một mảng, hãy chọn đoạn con mà sau khi sắp xếp đoạn này, được dãy sắp tăng dần

|  |
| --- |
| //   * Lấy 2 con trỏ chạy từ đầu và cuối, 2 con trỏ này đánh dấu vị trí đầu tiên mà dãy số gãy khúc (giả sử là trỏ l và r) * Trỏ l được cập nhật lại bằng vị trí từ đầu dãy đến left mà lớn hơn min[l, r] * Trỏ r được cập nhật lại bằng vị trí đầu tiên duyệt từ cuối dãy về r mà giá trị đó nhỏ hơn max trong đoạn [l, r] |

|  |
| --- |
| //code tham khảo  void sap\_day\_con(int a[], int n){  int l = 0, r = n - 1;  for(int i = 0; i < n - 1; i++){  if(a[i + 1] < a[i]){  l = i;  break;  }  }    for(int i = n - 1; i > 0; i--){  if(a[i - 1] > a[i]){  r = i;  break;  }  }    int min\_value = \*min\_element(a + l, a + r + 1);  int max\_value = \*max\_element(a + l, a + r + 1);  for(int i = 0; i <= l; i++){  if(a[i] > min\_value){  l = i;  break;  }  }    for(int i = n - 1; i >= r; i--){  if(a[i] < max\_value){  r = i;  break;  }  }    cout << l + 1 << " " << r + 1 << endl;;  } |

#### Biểu thức nhỏ nhất

* Cho dãy số nguyên không âm. Điền k dấu cộng và n – k – 1 dấu trừ vào giữa các số để được dãy lơn nhất

***Ý tưởng***

* Phần tử 1 vẫn luôn đặt dấu +
* Sắp xếp mảng từ phần tử thứ nhất trở đi (giảm dần)
* k phần tử lớn hơn sẽ đặt dấu +, còn lại đặt dấu –

|  |
| --- |
| //  int bieu\_thuc(int a[], int n, int k){  sort(a + 1, a + n, greater<int>());  long long res = a[0];  for(int i = 1; i < n; i++){  if(k >= 1)  res += a[i];  else  res -= a[i];  k--;  }  return res;  } |

#### Xếp hàng

* Có n khách đến làm thủ tục. Khách đến ở thời điểm ti và làm hết di thời gian. Tính thời điểm mà tất cả làm xong

***Gợi ý***

* Khách đến trước, làm trước. Sắp xếp theo thời gian đến trước

|  |
| --- |
| //  int main(){  int n; cin >> n;  pair<int, int>p[n];  for(int i = 0; i < n; i++){  cin >> p[i].first;  cin >> p[i].second;  }    sort(p, p + n);  int end\_time = -1;    for(int i = 0; i < n; i++){  end\_time = max(end\_time, p[i].first);  end\_time += p[i].second;  }    cout << end\_time;  } |

#### Cặp số gần 0

* Cho dãy sos, tính tổng 2 số bất kỳ trong dãy, sao cho tổng gần 0 nhất, nếu có nhiều cặp, lấy cặp có index nhỏ hơn

***Ý tưởng***

* Hai số có tổng gần bằng 0, nếu dãy có số âm và dương thù có thể là 1 số âm và dương, nếu chỉ toàn âm hoặc dương, thì là 2 số nhỏ nhất dãy
* Sắp xếp dãy tăng dần theo trị tuyệt đối, lưu lại index của các phần tử
* Cập nhật tổng tạm là tổng 2 số kề nhau. Nếu giá trị tuyệt đối của toonge tiếp theo nhỏ hơn, cập nhật tổng mới. index mới
* Nếu bằng và index mới nhỏ hơn, cập nhật tổng mới, index mới

|  |
| --- |
| //  bool cmp(pair<int, int>a, pair<int, int>b){  if(abs(a.first) != abs(b.first))  return abs(a.first) < abs(b.first);  return a.second < b.second;  }  void tim\_cap\_gan\_0(){  int n;  cin >> n;  vector<pair<int, int>>v(n);  /\*  Tao mang pair luu gia tri cua second: la gia tri mang  Luu gia tri cua first: La gia chi cua chi so  \*/  for(int i = 0; i < n; i++){  int x;  cin >> x;  v[i].first = x;  v[i].second = i;  }    sort(v.begin(), v.end(), cmp);  long long res, sum = INT\_MAX;  int idx, idx2;  for(int i = 1; i < n; i++){  int tmp = abs(v[i].first + v[i - 1].first);  if(tmp < sum){  sum = tmp;  res = v[i].first + v[i - 1].first;  idx = min(v[i].second, v[i - 1].second);  idx2 = max(v[i].second, v[i - 1].second);  }else if(tmp == sum){  if(idx > min(v[i].second, v[i - 1].second)){  res = v[i].first + v[i - 1].first;  idx = min(v[i].second, v[i - 1].second);  idx2 = max(v[i].second, v[i - 1].second);  }  }  }  cout <<"id = " << idx <<" " << idx2 << endl;  cout << "res = " << res << endl;  }  int main(){  int t;  cin >> t;  while(t--){  tim\_cap\_gan\_0();  }  } |

# Tìm kiếm

### binary\_search()

* Chỉ tìm trên mảng đã sắp xếp
* Là hàm có sẵn, dùng để tìm kiếm trên mảng sắp xếp
* Trả về true, false

|  |
| --- |
| //  int tk\_np(int a[], int l, int r, int x){  while(l <= r){  int m = (l + r)/2;  if(a[m] == x)  return 1;  else if(a[m] > x)  return tk\_np(a, l, m - 1, x);  else  return tk\_np(a, m + 1, r, x);  }  return 0;  } |

### Tìm vị trí đầu tiên

* Tìm liên tục về bên trái

|  |
| --- |
| //  int tk\_np\_dau\_tien(int a[], int l, int r, int x){  int res = -1;  while(l <= r){  int m = (l + r)/2;  if(a[m] == x){  res = m;  //tiep tuc tim ve ben trai  r = m - 1;  }  else if(a[m] > x)  //tim ve ben trai  r = m - 1;  else  //tim ve ben phai  l = m + 1;  }  return res;  } |

### Tìm vị trí cuối cùng

* Tìm liên tục về phải

|  |
| --- |
| //  int tk\_np\_cuoi\_cung(int a[], int l, int r, int x){  int res = -1;  while(l <= r){  int m = (l + r)/2;  if(a[m] == x){  res = m;  //tiep tuc tim ve ben phai  l = m + 1;  }  else if(a[m] > x)  //tim ve ben trai  r = m - 1;  else  //tim ve ben phai  l = m + 1;  }  return res;  } |

### lower\_bound

* Trả về vị trí đầu tiên, ***lớn hơn hoặc bằng k***

|  |
| --- |
| //  int a[n];  int k;  for(int i = 0; i < n; i++)  cin >> a[i];  cout << \*lower\_bound(a, a + n, k); |

* Nếu không có phần tử lớn hơn, hoặc bằng k, trả về vị trí cuối cùng

### upper\_bound()

* Trả về vị trí (iterator) trỏ đến phần tử đầu tiên ***lớn hơn k***

|  |
| --- |
| //  int a[n];  int k;  for(int i = 0; i < n; i++)  cin >> a[i];  cout << \*upper\_bound(a, a + n, k); |

* Nếu mảng toàn vị trí nhỏ hơn, hoặc bằng x, trả về vị trí cuối cùng

# Mảng

## Dãy con tăng

### Duyệt trâu

Đề bài: Cho một dãy số, đếm số lượng phần tử của dãy con tăng dài nhất

Giả sử có dãy số

[1, 2, 3, 1, 4, 8, 5]

* Dãy con tăng có thể là [1, 2, 3, 4, 5] hoặc [1, 2, 3, 4, 8]

***Ý tưởng***

* Giả sử ban đầu coi mỗi phần tử của dãy là một dãy con
* Ta tạo một mảng L[n] với L[i] là số lượng phần tử trong dãy con tăng của vị trí i
* Sau đó, tìm cách thêm phần tử này vào các dãy con phía trước nó. nếu phần tử a[i] (ở vị trí đang xét) lớn hơn phần tử a[j] nào (đứng trước a[i]) thì khi này L[i] sẽ được cập nhật bằng công thức max(l[j] + 1, L[i])
* Xét dãy [1, 2, 3, 1, 4, 8, 5]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bước |  | 1 | 2 | 3 | 1 | 4 | 8 | 5 |
| 1 | L[i] | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 |  | 1 | 2 | 3 | 1 | 4 | 5 | 5 |

* Tại bước 1, khởi tạo L[n]= 1
* Tại bước 2
  + Xét phấn tử 1, không có phần tử nào trước nó
  + Xét phần tử 2, 2>1 (1 phía trước), và 1 có L[0] = 1. Cập nhật L[1] (của phần tử 2) = max(L[0]+1, L[1]) = 2
  + Xét phần tử 3, (3>1) khi này L[2] = max(L[0]+1, L[2]) = max(2, 1) = 2.

Tiếp túc thấy 3>1, khi này L[2] = max(L[1]+1, L[2]) = max(3, 2) = 3

…

* Số phần tử của dãy con tăng dài nhất, là max của mảng L[n]

***Code demo***

|  |
| --- |
| #include <bits/stdc++.h>  #include <vector>  using namespace std;  int main(){  int n;  cin >> n;  vector <int>a(n);  for(int i = 0; i < n; i++){  cin >> a[i];  }    vector <int>L(n, 1);  for(int i = 0; i < n; i++){  for(int j = 0; j < i; j++){  if(a[i] > a[j]){  L[i] = max(L[i], L[j]+1);  }  }  }  cout <<\*max\_element(L.begin(), L.end());  } |

## Hai con trỏ

### Trộn mảng

* Trộn mảng đã được sắp xếp

|  |
| --- |
| //  vector<int> tron\_mang(vector<int>v1, vector<int>v2){  vector<int>v;  int i = 0, j = 0;  while(i < v1.size() && j < v2.size()){  if(v1[i] < v2[j]){  v.push\_back(v1[i]);  i++;  }else{  v.push\_back(v2[j]);  j++;  }  }    while(i < v1.size()){  v.push\_back(v1[i]);  i++;  }    while(j < v2.size()){  v.push\_back(v2[j]);  j++;  }    return v;  } |

### Tìm đoạn con lớn nhất có tổng bằng s

* Tìm độ dài đoạn con lớn nhất có tổng nhỏ hơn hoặc bằng s
* Duyệt trâu \_ O(n^2)

|  |
| --- |
| //  int doan\_con(int a[], int n, int s){  int res = -1;  for(int i = 0; i < n; i++){  int sum = 0;  for(int j = i; j < n; j++){  sum += a[j];  if(sum <= s)  res = max(res, j - i + 1);  }  }    return res;  } |

* Cách 2: Sử dụng 2 con trỏ

|  |
| --- |
| //  //su dung ky thuat 2 con tro  int day\_con(int a[], int n, int s){  int res = -1;  int l = 0, r;  int sum = 0;  for(r = 0; r < n; r++){  sum += a[r];  //dam bao day con co tong nho hon hoac bang s  //truoc khi cap nhat ket qua  while(sum > s){  sum -= a[l];  l++;  }  res = max(res, r - l + 1);  }  return res;  } |

# Độ phức tạp tính toán

* Tăng tốc độ đọc/ghi C++

|  |
| --- |
| //Đồng bộ cin, cout với scanf và printf  //Lưu ý: không dùng endl trong t/h này  ios\_base::sync\_with\_stdio(false);  cin.tie(NULL); |

## Cơ bản

### O(1)

* Các phép +, -, \*, /, mod
* phép gán

### O(log(n))

* binary search
* phân tích/kiểm tra số nguyên tố
* tìm kiếm/chèn/***xóa*** trong ***set***, ***map***

### O(n) – độ phức tạp tuyến tính

* nhập mảng
  + Nếu dùng for-loop lồng thì sẽ là tích của các vòng for

### O(nlogn)

### O(n^2)

# Sinh kế tiếp, quay lui

## Các thuật toán sinh cơ bản

### Sinh nhị phân

* Ý tưởng: Cộng 1 bit vào xâu nhị phân
* Chuyển 1 thành 0 và 0 thành 1. (gặp 0 thì dừng). Không thất bit 0, kết thuc thuật toán
* Cấu hình đầu: toàn số 0. Cấu hình cuối: toàn số 1

|  |
| --- |
| //  #include <bits/stdc++.h>  using namespace std;  int n, a[100], ok;  void ktao(){  for(int i = 1; i <= n; i++)  a[i] = 0;  }  void sinh(){  int i = n;  while(i >= 1 && a[i] == 1){  a[i--] = 0;  }  if(i == 0)  ok = 0;  else  a[i] = 1;  }  int main(){  ok = 1;  cin >> n;  ktao();  while(ok){  for(int i = 1; i <= n; i++)  cout << a[i];  cout << endl;  sinh();  }    } |

### Sinh tổ hợp

* Cấu hình đầu tiên: 1, 2, 3, …
* Cấu hình cuối cùng …n-2, n-1, n
* ***Điểm dừng: n-k+1***
* Điểm dừng vị trí: n-k+i

|  |
| --- |
| //  #include <bits/stdc++.h>  using namespace std;  int a[100], ok, n, k;  void init(){  for(int i = 1; i <= k; i++){  a[i] = i;  }  }  void next(){  int i = k;  while (i >= 1 && a[i] == n - k + i){  i--;  }  if(i == 0)  ok = 0;  else{  a[i]++;  for(int j = i + 1; j <= k; j++)  a[j] = a[j-1] + 1;  }  }  int main(){  ok = 1;  cin >> n >> k;  init();  while(ok){  for(int i = 1; i <= k; i++)  cout << a[i] << " ";    cout << endl;  next();  }  } |

### Sinh hoán vị

Ý tưởng

* Sinh ra cấu hình tiếp theo lớn hơn cấu hình phía trước
* Tìm vị trí gãy khúc đầu tiên (từ cuối về), đổi với số nhỏ nhất lớn hơn số ở vị trí gãy
* Đảo ngược dãy phía sau

|  |
| --- |
| //  int n, a[100], ok;  void init(){  for(int i = 1; i <= n; i++)  a[i] = i;  }  void next(){  //tim vi tri gay khuc dau tien (tu cuoi ve)  int i = n - 1;  while(i >= 1 && a[i] > a[i + 1]){  --i;  }  if(i == 0)  ok = 0;  else{  //tim a[j] nho nhat nhung lon hon a[i]  int j = n;  while(a[i] > a[j]){  --j;  }  swap(a[i], a[j]);  //lat nguoc doan phia sau  // int l = i + 1, r = n;  // while(l < r){  // swap(a[l], a[r]);  // l++;  // r--;  // }    //ham co san lat nguoc: reverse  reverse(a + i + 1, a + n + 1);  }  }  int main(){  cin >> n;  init();  ok = 1;  while(ok){  for(int i = 1; i <= n; i++)  cout << a[i] << " ";  cout << endl;  next();  }  } |

|  |
| --- |
| //Hàm có sẵn  int a[100], n, ok;  void init(){  for(int i = 0; i < n; i++)  a[i] = i + 1;  }  int main(){  cin >> n;  ok = 1;  init();  do{  for(int i = 0; i < n; i++)  cout << a[i] << " ";  cout << endl;  }while(next\_permutation(a, a + n));  } |

* Sinh ngược

|  |
| --- |
| //Sinh hoán vị ngược  int a[100], n, ok;  void init(){  for(int i = 0; i < n; i++){  a[i] = n - i;  }  }  int main(){  cin >> n;  init();  do{  for(int i = 0; i < n; i++)  cout << a[i] << " ";  cout << endl;  }while(prev\_permutation(a, a + n));  } |

### Sinh phân hoạch

* Biểu diễn 1 số bằng tổng các số nhỏ hơn nó

## Các thuật toán quay lui cơ bản

Mã giả

|  |
| --- |
| //  back\_track(int i){  for(j = <kha\_nang 1>; j <= n; j++){  if(<chap\_nhan\_kha\_nang>j){  x[i] = <kha\_nang\_j>;  if(i == n)  result();  else  back\_track(i + 1);  }  }  } |

### Quay lui nhị phân

|  |
| --- |
| //  int a[100], n;  void sinh\_np(int i){  cout <<"De qui bit thu " << i << endl;  for(int j = 0; j <= 1; j++){  a[i] = j;  cout << "Gan cho bit thu " << i << " gia tri = " << j << endl;  if(i == n){  //result();  for(int i = 1; i <= n; i++)  cout << a[i];  cout << endl;  cout << "Quay lui song voi nhanh thu " << i << endl;  }  else  sinh\_np(i + 1);  }  }  int main(){  cin >> n;  sinh\_np(1);  } |

A diagram of a mathematical equation

Description automatically generated

### Quay lui tổ hợp

* Cận dưới: a[i-1]+1
* Cận trên: n – k + i

|  |
| --- |
| //  int a[100], n, k;  void sinh\_np(int i){  for(int j = a[i - 1] + 1; j <= n - k + i; j++){  a[i] = j;  if(i == k){  for(int i = 1; i <= k; i++)  cout << a[i];  cout << endl;  }  else  sinh\_np(i + 1);  }  }  int main(){  cin >> n >> k;  a[0] = 0;  sinh\_np(1);  } |

A diagram of a graph

Description automatically generated

### Sinh hoán vị

* Sử dụng mảng đánh dấu xem phần tử đó đã được sử dụng chưa

|  |
| --- |
| //  #include <iostream>  using namespace std;  int a[100], n;  bool used[100];  void sinh\_hv(int i){  for(int j = 1; j <= n; j++){  //\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  if(used[j] == false){  a[i] = j;  used[j] = true;    if(i == n){  //result();  for(int i = 1; i <= n; i++)  cout << a[i];  cout << endl;  }  else  sinh\_hv(i + 1);    //\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  used[j] = false;  }    }  }  int main(){  cin >> n;  memset(used, false, sizeof(used));  sinh\_hv(1);  } |

### Bài toán quân hậu

* Giả sử đặt môi quân hậu tại một hàng, ta xét xem cột, đường chéo 1, đường chéo 2 tại vị trí định đặt có khả thi không

|  |
| --- |
| //Mã giả  void try(int i){  for(int j = 1; j <= n; j++){  if(a[i] && xuoi[i - j + n] && nguoc [i + j - 1]){  x[i] = j;  a[i] = false;  xuoi[i - j + n] = false;  nguoc[i + j - 1] = false;  if(i == n)  result();  else  try(i + 1);  a[j] = true;  xuoi [i - j + n] = true;  nguoc[i + j - 1] = true;  }  }  } |

# Ngăn xếp, hàng đợi

## Queue

Ví dụ có các phần tử : 1, 2, 3, 5, 6

* Phần tử được săps xếp tăng dần. Khi lấy phần tử top, sẽ luôn là phần tử lớn nhất
* Một số thao tác chính
  + push()
  + pop()
  + top()

|  |
| --- |
| //  priority\_queue<int>pq;  priority\_queue<int, vector<int>, greater<int>>pq; |

### Hàng đợi

* Vào trước, ra trước
* Phần tử, được chèn vào trước, sẽ lấy ra trước

# Đồ thị

## Khái niệm cơ bản

Đồ thị G<V, E>

* Là đồ thị được biểu diễn bởi 1 tập đỉnh và 1 tập cạnh (V; là đỉnh, E: cạnh)
* Có 2 loại: có hướng, vô hướng
* Có thể chia: Đơn đồ thị, đa đồ thị
* Khuyên: Cạnh đi từ nó đến chính nó
* Cạnh
  + Không hướng: cạnh
  + Có hướng: cung

Tổng các bậc = 2 \* n(Với n là số cạnh)

### Đồ thị liên thông

* 2 đỉnh bất kỳ luôn có đường đi

### Đồ thị vô hướng

* Ma trận kề biểu diễn sẽ là ma trận đối xứng

### Đồ thị có hướng

* Bán bậc ra; count trên hàng
* Bán bậc vào: count trên cột

### Ma trận trọng số

### Danh sách cạnh

## Các phương án biểu diễn

### Ma trận kề

0 1 0 0 0

1 0 1 1 1

0 1 0 1 1

0 1 1 0 0

0 1 1 0 0

### Danh sách kề

1: 2

2: 1 3 4 5

3: 2 4 5

4: 2 3

5: 2 3

### Danh sách cạnh

1 2

2 3

2 4

2 5

3 4

3 5

## Các thuật toán duyệt đồ thị cơ bản

### DFS (Tìm theo chiêu sâu)

|  |
| --- |
| //  DFS(u){//u là đỉnh bắt đầu duyệt  <thăm đỉnh u>; //duyệt đỉnh u  chuaxet[u] = false; //xác nhận u đã được duyệt  for(v thuoc Ke(u)){  if(chuaxet[v])  DFS(v); //duyệt v nếu nó chưa được xét  }  } |

* Cài đặt với ma trận kề

|  |
| --- |
| //  int a[1001][1001];  int n;  void dfs(int u){  cout << u <<" ";  for(int i = 1; i <= n; i++){  //duyet cac cot xem co canh noi voi i khong  if(a[u][i] == 1){  //Dat gia trij = 0 cho cac cap noi da xet  a[u][i] = a[i][u] = 0;  dfs(i);  }  }  }  int main(){  cin >> n;  for(int i = 1; i <= n; i++){  for(int j = 1; j <= n; j++)  cin >> a[i][j];  }    dfs(1);  }  /\*  10  0 1 0 0 1 0 0 1 0 0  1 0 1 1 0 0 0 0 0 0  0 1 0 0 0 0 0 0 0 0  0 1 0 0 0 0 0 0 0 0  1 0 0 0 0 1 1 0 0 0  0 0 0 0 1 0 0 0 0 0  0 0 0 0 1 0 0 0 0 0  1 0 0 0 0 0 0 0 1 1  0 0 0 0 0 0 0 1 0 0  0 0 0 0 0 0 0 1 0 0  \*/ |

### BFS(Tìm theo chiều rộng)

|  |
| --- |
| //  1. khởi tạo  quêu = rỗng;  push(queue, u); chuaxet[u] = false;  2. Lặp  while(queue != rỗng){  s = pop(queue){  Thăm đỉnh s  for(t thuộc ke(s)){  if(chuaxet[t]){  push(queue, t);  chuaxet[t] = false;  }  }  }  }  3. Trả lại kết quả  return <Tập đỉnh đã duyệt> |

### Ứng dụng

#### Kiểm tra liên thông

* Đếm số mảnh

#### Kiểm tra liên thông mạnh (1 đỉnh đi qua tất cả đỉnh còn lại)

#### Tìm đường đi

* Sử dụng thêm một mảng để lưu vết
* Tìm đường đi với dfs(tìm sâu)

|  |
| --- |
| //  bool used[1001];  vector<int>adj[1001];  int parent[1001];  int n, m, x, y;  void dfs(int u){  used[u] = true;  for(int it: adj[u]){  if(used[it] == false){  parent[it] = u;  dfs(it);  }  }  }  int main(){  int s, e;  cin >> n >> m >> s >> e;  while(m--){  cin >> x >> y;  adj[x].push\_back(y);  adj[y].push\_back(x);  }  dfs(s);    if(used[e] == false){  cout << -1 << endl;  }else{  vector<int>path;  while(e != s){  path.push\_back(e);  e = parent[e];  }  path.push\_back(e);  reverse(path.begin(), path.end());    for(int it: path)  cout << it << " ";  }  }  /\*  6 9 1 6  1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6  \*/ |

* Tìm đường đi với bfs(tìm rộng)

|  |
| --- |
| //  bool used[1001];  vector<int> adj[1001];  int parent[1001];  int n, m, s, e;  void bfs(int u){  queue<int>q;  q.push(u);  used[u] = true;    while(!q.empty()){  int t = q.front();  q.pop();  for(int it: adj[t]){  if(used[it] == false){  q.push(it);  used[it] = true;  parent[it] = t;  }  }  }  }  int main(){  cin >> n >> m >> s >> e;  while(m--){  int x, y;  cin >> x >> y;  adj[x].push\_back(y);  adj[y].push\_back(x);  }    bfs(s);  if(used[e] == false)  cout << -1 << endl;  else{  vector<int>path;  while(e != s){  path.push\_back(e);  e = parent[e];  }  path.push\_back(e);  reverse(path.begin(), path.end());  for(int it: path)  cout << it << " ";  }  }  /\*  6 9 1 6  1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6  \*/ |

* Trên đồ thị không trọng số => bfs cho đường đi ngắn nhất

#### DFS và BFS trên grid

Dfs trên grid

|  |
| --- |
| //  int n, m, a[501][501];  int dx[8] = {0, -1, -1, -1, 0, 1, 1, 1};  int dy[8] = {-1, -1, 0, 1, 1, 1, 0, -1};  void dfs(int i, int j){  a[i][j] = -1;  for(int k = 0; k < 8; k++){  int i1 = i + dx[k];  int j1 = j + dy[k];  if(i1 >= 1 && i1 <= n && j1 >= 1 && j1 <= m && a[i1][j1] == 1){  dfs(i1, j1);  }  }  }  int main(){  int t;  cin >> t;  while(t--){  cin >> n >> m;  for(int i = 1; i <= n; i++){  for(int j = 1; j <= m; j++)  cin >> a[i][j];  }    int cnt = 0;  for(int i = 1; i <= n; i++){  for(int j = 1; j <= m; j++){  if(a[i][j] == 1){  cnt++;  dfs(i, j);  }  }  }      cout << endl << "res = " << cnt << endl;  for(int i = 1; i <= n; i++){  for(int j = 1; j <= m; j++)  cout << a[i][j] << " ";  cout <<"\n";  }  }  }  /\*  1  5 5  1 1 0 0 0  0 1 0 0 1  1 0 0 1 1  0 0 0 0 0  1 0 1 0 1  //  1  5 5  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  \*/ |

bfs trên grid

|  |
| --- |
| //  int n, m, a[501][501];  int dx[8] = {0, -1, -1, -1, 0, 1, 1, 1};  int dy[8] = {-1, -1, 0, 1, 1, 0, 1, -1};  void bfs(int i, int j){  queue<pair<int, int>>q;  q.push({i, j});  a[i][j] = -1;    while(!q.empty()){  pair<int, int> t = q.front();  q.pop();  for(int k = 0; k < 8; k++){  int i1 = t.first + dx[k];  int j1 = t.second + dy[k];  if(i1 >= 1 && i1 <= n && j1 >= 1 && j1 <= m && a[i1][j1] == 1){  q.push({i1, j1});  a[i1][j1] = -1;  }  }  }  }  void dem(){  int res = 0;  for(int i = 1; i <= n; i++){  for(int j = 1; j <= m; j++){  if(a[i][j] == 1){  bfs(i, j);  res ++;  }  }  }  cout << endl;  cout <<"So dao = " << res << endl;  for(int i = 1; i <= n; i++){  for(int j = 1; j <= m; j++)  cout << a[i][j] << " ";  cout << endl;  }  }  int main(){  int t;  cin >> t;  while(t--){  cin >> n >> m;  for(int i = 1; i <= n; i++){  for(int j = 1; j <= m; j++)  cin >> a[i][j];  }  dem();  }  }  /\*  1  5 5  1 1 0 0 0  0 1 0 0 1  1 0 0 1 1  0 0 0 0 0  1 0 1 0 1  //  1  5 5  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  \*/ |

Đường đi 2 đỉnh grid

|  |
| --- |
| //  int n, m, a[501][501], d[501][501];  int s1, s2, t1, t2;  int dx[8] = {0, -1, -1, -1, 0, 1, 1, 1};  int dy[8] = {-1, -1, 0, 1, 1, 0, 1, -1};  void bfs(int i, int j){  queue<pair<int, int>>q;  q.push({i, j});  a[i][j] = -1;  d[i][j] = 0;    while(!q.empty()){  pair<int, int> t = q.front();  q.pop();  for(int k = 0; k < 8; k++){  int i1 = t.first + dx[k];  int j1 = t.second + dy[k];  if(i1 >= 1 && i1 <= n && j1 >= 1 && j1 <= m && a[i1][j1] == 1){  q.push({i1, j1});  a[i1][j1] = -1;  d[i1][j1] = d[t.first][t.second] + 1;  }  }  }  }  void solve(){  bfs(s1, s2);  cout << d[t1][t2] << endl;  }  int main(){  int t;  cin >> t;  while(t--){    cin >> n >> m >> s1 >> s2 >> t1 >> t2;  for(int i = 1; i <= n; i++){  for(int j = 1; j <= m; j++)  cin >> a[i][j];  }  solve();  }  }  /\*  1  5 5 0 0 3 1  1 1 0 0 0  0 1 0 0 1  1 0 0 1 1  0 1 0 0 0  1 0 1 0 1  //  1  5 5  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  \*/ |

#### Đỉnh trụ

* Đỉnh u là đỉnh trụ của một đồ thị vô hướng, nếu loại bỏ đỉnh đó sẽ làm tăng thành phần liên thông của đồ thị
* Đồ thị liên thông

|  |
| --- |
| //  int n, m , x, y;  vector<int>adj[1001];  bool used[1001];  void dfs(int u){  used[u] = true;  for(int it: adj[u]){  if(!used[it])  dfs(it);  }  }  int main(){  cin >> n >> m;  while(m--){  cin >> x >> y;  adj[x].push\_back(y);  adj[y].push\_back(x);  }  for(int i = 1; i <= n; i++){  memset(used, false, sizeof(used));  used[i] = true;  if(i == 1)  dfs(2);  else  dfs(1);  int ok = 1;  for(int i = 1; i <= n; i++){  if(used[i] == false){  ok = 0;  break;  }  }  if(!ok)  cout << i << " ";  }  }  /\*  5 5  1 2 1 3 2 3 2 5 3 4  \*/ |

* Đồ thị không liên thông

|  |
| --- |
| //  int n, m , x, y;  vector<int>adj[1001];  bool used[1001];  void dfs(int u){  used[u] = true;  for(int it: adj[u]){  if(!used[it])  dfs(it);  }  }  int main(){  cin >> n >> m;  while(m--){  cin >> x >> y;  adj[x].push\_back(y);  adj[y].push\_back(x);  }  int cnt = 0;  memset(used, false, sizeof(used));  for(int i = 1; i <= n; i++){  if(!used[i]){  cnt ++;  dfs(i);  }  }  cout << cnt << endl;      for(int i = 1; i <= n; i++){  memset(used, false, sizeof(used));  used[i] = true;  int dem = 0;  for(int i = 1; i <= n; i++){  if(!used[i]){  dfs(i);  dem ++;  }  }  if(dem > cnt)  cout << i << " ";  }  }  /\*  5 5  1 2 1 3 2 3 2 5 3 4  \*/ |

#### Cạnh cầu

* Loại bỏ cạnh cầu làm đồ thị tăng thành phần liên thông

## Một số thuật toán khác

#### Sắp xếp topo

|  |
| --- |
| //  bool used[1001];  vector<int>adj[1001];  vector<int>topo;  int n, m, x, y;  void dfs(int u){  used[u] = true;  for(int it: adj[u]){  if(!used[it])  dfs(it);  }  topo.push\_back(u);  }  int main(){  cin >> n >> m;  while(m--){  cin >> x >> y;  adj[x].push\_back(y);  }  dfs(1);  reverse(topo.begin(), topo.end());  for(int it: topo)  cout << it <<" ";  }  /\*  6 5  1 2  2 3  2 4  1 5  3 6  \*/ |

#### Thuật toán kahn

* Thuật toán kahn không hoạt động neeusddoof thị tạo thành 1 chu trình

|  |
| --- |
| //  vector<int>adj[1001];  vector<int>topo;  int degree[1001] = {0};  int n, m, x, y;  void kahn(){  queue<int>q;  for(int i = 1; i <= n; i++){  if(degree[i] == 0)  q.push(i);  }    while(!q.empty()){  int v = q.front();  q.pop();  topo.push\_back(v);  for(int it: adj[v]){  degree[it]--;  if(degree[it] == 0)  q.push(it);  }  }  }  int main(){  cin >> n >> m;  while(m--){  cin >> x >> y;  adj[x].push\_back(y);  degree[y]++;  }  kahn();  for(int it: topo)  cout << it << " ";  }  /\*  7 7  1 2  1 5  2 3  2 4  3 6  4 6  7 6  \*/ |

#### Chu trình vô hướng

* Kiểm tra chu trình dfs

|  |
| --- |
| //  #include <bits/stdc++.h>  #include <vector>  using namespace std;  int visited[1001];  int parent[1001];  bool dfs(int u){  visited[u] = true;  for(int x: adj[u]){  if(visited[x]){  if(x != parent[u])  return true;  }else{  parent[x] = u;  if(dfs(x))  return true;  }  }  return false;  } |

* Kiểm tra chu trình bfs

|  |
| --- |
| //  #include <bits/stdc++.h>  #include <vector>  using namespace std;  int visited[1001];  int parent[1001];  bool bfs(int u){  queue<int>q;  q.push(u);  visited[u] = true;    while(!q.empty()){  int v = q.front();  q.pop();  for(int x: adj[v]){  if(visited[x]){  if(x != parent[v])  return true;  }else{  q.push(u);  visited[u];  parent[x] = v;  }  }  }  } |

#### Chu trình có hướng

* Dfs

|  |
| --- |
| //  int n, m, x, y, cnt = 0;  bool used[1001];  vector<int> adj[1001];  int color[1001] = {0};  bool dfs(int u){  color[u] = 1;  for(int v: adj[u]){  if(color[v] == 0){  if(dfs(v))  return true;  }else if(color[v] == 1){  return true;  }  }  color[u] = 2;  return false;  }  int main(){  cin >> n >> m;  while(m--){  cin >> x >> y;  adj[x].push\_back(y);  }    int ok = 0;  for(int i = 1; i <= n; i++){  if(color[i] == 0){  if(dfs(i)){  ok = 1;  break;  }  }  }  if(ok)  cout <<"YES\n";  else  cout <<"NO\n";  } |

* Kahn

|  |
| --- |
| //  int n, m, x, y, cnt = 0;  bool used[1001];  vector<int> adj[1001];  int color[1001] = {0};  int degree[1001] = {0};  void kahn(){  queue<int>q;  for(int i = 1; i <= n; i++){  if(degree[i] == 0)  q.push(i);  }    int cnt = 0;    while(!q.empty()){  int v = q.front();  q.pop();  cnt++;  cout << v << "\_";  for(int x: adj[v]){  degree[x] --;  if(degree[x] == 0){  q.push(x);  }  }  }    cout << cnt << endl;  if(cnt == n){  cout <<"NO\n";  }  else  cout <<"YES\n";  }  int main(){  cin >> n >> m;  while(m--){  cin >> x >> y;  adj[x].push\_back(y);  degree[y] ++;  }  cout << "\n\n";  for(int i = 1; i <= n; i++){  cout << i << " " << degree[i] << endl;  }  kahn();  }  /\*  5 6  1 2  2 3  3 1  2 4  3 4  1 5  \*/ |

#### Đồ thị 2 phía

|  |
| --- |
| //bfs  int n, m, x, y, cnt = 0;  bool used[1001];  vector<int> adj[1001];  int color[1001] = {0};  int degree[1001] = {0};  bool bfs(int u){  queue<int>q;  q.push(u);  color[u] = 1;  while(!q.empty()){  int v = q.front();  q.pop();  for(int x: adj[v]){  if(color[x] == color[v])  return false;  else if(color[x] == 0){  q.push(x);  color[x] = 3 - color[v];  }  }  }  return true;  }  int main(){  cin >> n >> m;  while(m--){  cin >> x >> y;  adj[x].push\_back(y);  adj[y].push\_back(x);  }  cout << bfs(1);  } |

## Cây khung nhỏ nhất

### Union and find

|  |
| --- |
| //  int parent[1001];  int n, m, x, y;  int size[1001];  int find(int u){  // while(u != parent[u])  // u = parent[u];  // return u;  //nen duong  if(u == parent[u])  return u;  return parent[u] = find(parent[u]);  }  bool Union(int x, int y){  x = find(x);  y = find(y);  // if(x < y)  // parent[y] = x;  // else  // parent[x] = y;    //toi uu theo size  if(x == y)  return false;  if(x != y){  if(size(x) >= size(y)){  parent[y] = x;  size[x] += size[y];  }else{  parent[x] = y;  size[y] += size[x];  }  }  return true;  }  int main(){  cin >> n >> m;  for(int i = 1; i <= n; i++){  parent[i] = i;  size[i] = 1;  }    while(m--){  cin >> x >> y;  Union(x, y);  }    cout << endl;    for(int i = 1; i <= n; i++)  cout << i << " " << parent[i] <<"\n";  }  /\*  7 6  1 2  2 3  2 4  3 5  5 6  6 7  \*/ |

### Cây

* Một cây có n đỉnh thì chỉ có n – 1 cạnh và không có chu trình

### Kruskal

A screenshot of a computer

Description automatically generated

## Đường đi ngắn nhất

### Dijistra

* Tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại trên đồ thị
* Không áp dụng với đồ thị có trọng số âm

A screenshot of a math problem

Description automatically generated

|  |
| --- |
| //  int n, m;  vector<pair<int, int>>adj[1001];  vector<int>d;  void dijkstra(int s){  d.assign(n + 1, INT\_MAX);  d[s] = 0;  priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>>q;  q.push({0, s});  while(!q.empty()){  pair<int, int>t = q.top();  q.pop();  int u = t.second;  int len = t.first;  if(len > d[u])  continue;  for(auto it: adj[u]){  int v = it.first, w = it.second;  if(d[v] > d[u] + w){  d[v] = d[u] + w;  q.push({d[v], v});  }    }  }    cout << "\n\n";  for(int i = 1; i <= n; i++){  cout << i << " " << d[i] << endl;  }  }  int main(){  cin >> n >> m;  while(m--){  int x, y, w;  cin >> x >> y >> w;  adj[x].push\_back({y, w});  adj[y].push\_back({x, w});  }  dijkstra(1);  }  /\*  9 12  1 2 4  1 8 8  2 3 8  2 8 11  3 4 7  3 6 4  3 9 2  4 5 9  4 6 14  5 6 10  6 7 2  6 9 6  \*/ |

### Bellman Ford

* Không áp dụng với đồ thị có chu trình âm
* Có thể áp dụng khi trọng số âm

A white background with black text

Description automatically generated

|  |
| --- |
| //  vector<pair<int, int>>adj[1001];  int n, m;  void inp(){  cin >> n >> m;  while(m--){  int x, y, w;  cin >> x >> y >> w;  adj[x].push\_back({y, w});  }  }  void bellmanford(int u){  vector<int>d(n + 1, 1e9);  d[u] = 0;  for(int i = 1; i <= n - 2; i++){  for(int u = 1; u <= n; u++){  for(auto it: adj[u]){  int v = it.first, c = it.second;  d[v] = min(d[v], d[u] + c);  }  }  }  cout << endl << endl;  for(int i = 1; i <= n; i++)  cout << d[i] << " ";  }  int main(){  inp();  bellmanford(1);  }  /\*  5 9  1 2 1  1 5 3  2 5 8  2 3 3  2 4 3  3 4 1  4 5 -5  4 3 2  5 4 4  \*/ |

### Floyd

## Chu trình

### Chu trình Ơ-le

* Đi qua mỗi cạnh 1 lần duy nhất
  + **Vô hướng**

***Chu trình***

Tất cả bậc của đỉnh là bậc chẵn có chu trình Ơ-le

***Đường đi***

Tất cả bậc của đỉnh là bậc chẵn và có 2 bậc lẻ. Đường đi sẽ bắt đầu tại 1 trong 2 điểm này

* + **Có hướng**

Tất cả các đỉnh có bán bậc ra bằng bán bậc vào