CẤU TRÚC DỮ LIỆU HÀNG ĐỢI 1

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08001/top)

Ban đầu cho một queue rỗng. Bạn cần thực hiện các truy vấn sau:

1. Trả về kích thước của queue
2. Kiểm tra xem queue có rỗng không, nếu có in ra “YES”, nếu không in ra “NO”.
3. Cho một số nguyên và đẩy số nguyên này vào cuối queue.
4. Loại bỏ phần tử ở đầu queue nếu queue không rỗng, nếu rỗng không cần thực hiện.
5. Trả về phần tử ở đầu queue, nếu queue rỗng in ra -1.
6. Trả về phần tử ở cuối queue, nếu queue rỗng in ra -1.

**Dữ liệu vào**

Dòng đầu tiên chứa số nguyên T là số bộ dữ liệu, mỗi bộ dữ theo dạng sau.

Dòng đầu tiên chứa số nguyên n - lượng truy vấn (1 ≤ n ≤ 1000)

N dòng tiếp theo, mỗi dòng sẽ ghi loại truy vấn như trên, với truy vấn loại 3 sẽ có thêm một số nguyên, không quá 106.

**Kết quả:** In ra kết quả của các truy vấn..

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1  14  3 1  3 2  3 3  5  6  4  4  4  4  4  3 5  3 6  5  1 | 1  3  5  2 |

#include<bits/stdc++.h>

using namespace std;

int main(){

int t; cin >> t;

while (t--){

int n; cin >> n;

queue<int> q;

while (n--){

int k; cin >> k;

if (k == 1)

cout << q.size() << endl;

if (k == 2){

if (q.empty()) cout << "YES" << endl;

else cout << "NO" << endl;

}

if (k == 3){

int x; cin >> x;

q.push(x);

}

if (k == 4){

if (!q.empty())

q.pop();

}

if (k == 5){

if (q.empty()) cout << "-1" << endl;

else cout << q.front() << endl;

}

if (k == 6){

if (q.empty()) cout << "-1" << endl;

else cout << q.back() << endl;

}

}

}

}

CẤU TRÚC DỮ LIỆU HÀNG ĐỢI 2

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08002/top)

Yêu cầu bạn xây dựng một queue với các truy vấn sau đây:

“PUSH x”: Thêm phần tử x vào cuối của queue (0 ≤ x ≤ 1000).

“PRINTFRONT”: In ra phần tử đầu tiên của queue. Nếu queue rỗng, in ra “NONE”.

“POP”: Xóa phần tử ở đầu của queue. Nếu queue rỗng, không làm gì cả.

**Dữ liệu vào:**

Dòng đầu tiên là số lượng truy vấn Q (Q ≤ 100000).

Mỗi truy vấn có dạng như trên.

**Kết quả:**

Với mỗi truy vấn “PRINT”, hãy in ra phần tử đầu tiên của queue. Nếu queue rỗng, in ra “NONE”.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 9  PUSH 1  PUSH 2  POP  PRINTFRONT  PUSH 3  PRINTFRONT  POP  POP  PRINTFRONT | 2  2  NONE |

#include<bits/stdc++.h>

using namespace std;

int main(){

int n; cin >> n;

queue<int> q;

while (n--){

string str; cin >> str;

if (str == "PUSH"){

int x; cin >> x;

q.push(x);

}

if (str == "POP"){

if (!q.empty())

q.pop();

}

if (str == "PRINTFRONT")

if (!q.empty())

cout << q.front() << endl;

else cout << "NONE" << endl;

}

}

HÀNG ĐỢI HAI ĐẦU (DEQUEUE)

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08003/top)

Yêu cầu bạn xây dựng một hàng đợi hai đầu với các truy vấn sau đây:

“PUSHFRONT x”: Thêm phần tử x vào đầu của dequeue (0 ≤ x ≤ 1000).

“PRINTFRONT”: In ra phần tử đầu tiên của dequeue. Nếu dequeue rỗng, in ra “NONE”.

“POPFRONT”: Xóa phần tử đầu của dequeue. Nếu dequeue rỗng, không làm gì cả.

“PUSHBACK x”: Thêm phần tử x vào cuối của dequeue (0 ≤ x ≤ 1000).

“PRINTBACK”: In ra phần tử cuối của dequeue. Nếu dequeue rỗng, in ra “NONE”.

“POPBACK”: Xóa phần tử cuối của dequeue. Nếu dequeue rỗng, không làm gì cả.

**Dữ liệu vào:**

Dòng đầu tiên là số lượng truy vấn Q (Q ≤ 100000).

Mỗi truy vấn có dạng như trên.

**Kết quả:**

Với mỗi truy vấn “PRINTFRONT” và “PRINTBACK”, hãy in ra kết quả trên một dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10  PUSHBACK 1  PUSHFRONT 2  PUSHBACK 3  PRINTFRONT  POPFRONT  PRINTFRONT  POPFRONT  PRINTBACK  POPFRONT  PRINTBACK | 2  1  3  NONE |

#include<bits/stdc++.h>

using namespace std;

int main(){

int n; cin >> n;

deque<int> q;

while (n--){

string str; cin >> str;

if (str == "PUSHFRONT"){

int x; cin >> x;

q.push\_front(x);

}

if (str == "POPFRONT"){

if (!q.empty())

q.pop\_front();

}

if (str == "PRINTFRONT")

if (!q.empty())

cout << q.front() << endl;

else cout << "NONE" << endl;

if (str == "PUSHBACK"){

int x; cin >> x;

q.push\_back(x);

}

if (str == "PRINTBACK"){

if (!q.empty())

cout << q.back() << endl;

else cout << "NONE" << endl;

}

if (str == "POPBACK"){

if (!q.empty())

q.pop\_back();

}

}

}

SỐ NHỊ PHÂN TỪ 1 ĐẾN N

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08005/top)

Cho số tự nhiên n. Hãy in ra tất cả các số nhị phân từ 1 đến n.

**Input:**

* Dòng đầu tiên ghi lại số lượng test T (T≤100).
* Mỗi test là một số tự nhiên n được ghi trên một dòng (n≤10000).

**Output:**

* Đưa ra kết quả mỗi test trên một dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  2  5 | 1 10  1 10 11 100 101 |

#include<bits/stdc++.h>

using namespace std;

void solve(int n){

queue<string> q;

q.push("1");

while (n--){

string s1 = q.front();

q.pop();

cout << s1 << " ";

string s2 = s1;

q.push(s1.append("0"));

q.push(s2.append("1"));

}

cout << endl;

}

int main(){

int t; cin >> t;

while (t--){

int n; cin >> n;

solve(n);

}

}

SỐ 0 VÀ SỐ 9

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08006/top)

Cho số tự nhiên N. Hãy tìm số nguyên dương X nhỏ nhất được tạo bởi số 9 và số 0 chia hết cho N. Ví dụ với N = 5 ta sẽ tìm ra  X = 90.

**Input:**

* Dòng đầu tiên ghi lại số lượng test T (T≤100).
* Những dòng kế tiếp mỗi dòng ghi lại một test. Mỗi test là một số tự nhiên N được ghi trên một dòng (N≤100).

**Output:**

* Đưa ra theo từng dòng số X nhỏ nhất chia hết cho N tìm được .

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  5  7 | 90  9009 |

#include<bits/stdc++.h>

using namespace std;

void solve(int n){

queue<long long> q;

q.push(9);

while (!q.empty()){

long long x = q.front();

q.pop();

if (x%n == 0){

cout << x << endl;

break;

}

q.push(x\*10);

q.push(x\*10+9);

}

}

int main(){

int t; cin >> t;

while(t--){

int n; cin >> n;

solve(n);

}

}

SỐ BDN 1

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08007/top)

Ta gọi số nguyên dương K là một số BDN nếu các chữ số trong K chỉ bao gồm các 0 hoặc 1 có nghĩa. Ví dụ số K = 1, 10, 101. Cho số tự nhiên N (N<263). Hãy cho biết có bao nhiêu số BDN nhỏ hơn N. Ví dụ N=100 ta có 4 số BDN bao gồm các số: 1, 10, 11, 100.

**Input:**

* Dòng đầu tiên ghi lại số tự nhiên T là số lượng Test;
* T dòng kế tiếp mỗi dòng ghi lại một bộ Test. Mỗi test là một số tự nhiên N.

**Output:**

* Đưa ra kết quả mỗi test theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  10  100  200 | 2  4  7 |

#include<bits/stdc++.h>

using namespace std;

#include<bits/stdc++.h>

using namespace std;

void solve(long long n){

queue<long long> q;

q.push(1);

long long res = 0;

while (!q.empty()){

long long x = q.front();

q.pop();

if (x <= n){

res++;

q.push(x\*10);

q.push(x\*10+1);

}

}

cout << res << endl;

}

int main(){

int t; cin >> t;

while (t--){

long long n; cin >> n;

solve(n);

}

}

SỐ BDN 2

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08008/top)

Ta gọi số nguyên dương K là một số BDN nếu các chữ số trong K chỉ bao gồm các 0 hoặc 1 có nghĩa. Ví dụ số K =  101 là số BDN, k=102 không phải là số BDN.

Số BDN của N là số P =M\*N sao cho P là số BDN. Cho số tự nhiên N (N<1000), hãy tìm số BDN nhỏ nhất của N.

**Ví dụ**. Với N=2, ta tìm được số BDN của N là P = 5\*2=10. N = 17 ta tìm được số BDN của 17 là P = 653\*17=11101.

**Input:**

* Dòng đầu tiên ghi lại số tự nhiên T là số lượng Test;
* T dòng kế tiếp mỗi dòng ghi lại một bộ Test. Mỗi test là một số tự nhiên N.

**Output:**

                        Đưa ra kết quả mỗi test theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  2  12  17 | 10  11100  11101 |

#include<bits/stdc++.h>

typedef long long ll;

using namespace std;

ll mod(string str, ll n){

ll r = 0;

for (ll i = 0; i < str.length(); i++){

r = r\*10 + (str[i]-'0');

r %= n;

}

return r;

}

void solve(long long n){

queue<string> q;

q.push("1");

while (!q.empty()){

string str = q.front();

q.pop();

long long r = mod(str, n);

if (r == 0){

cout << str << endl;

break;

}

q.push(str+"0");

q.push(str+"1");

}

}

int main(){

int t; cin >> t;

while (t--){

long long n; cin >> n;

solve(n);

}

}

SỐ LỘC PHÁT 1

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08017/top)

Một số được gọi là lộc phát nếu chỉ có 2 chữ số 6 và 8. Cho số tự nhiên N. Hãy liệt kê các số lộc phát có không quá N chữ số.

**Input:**

* Dòng đầu tiên ghi lại số tự nhiên T là số lượng bộ test (T<10);
* T dòng kế tiếp mỗi dòng ghi số N (1<N<15).

**Output:**

* In ra đáp án theo thứ tự giảm dần.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  2  3 | 88 86 68 66 8 6  888 886 868 866 688 686 668 666 88 86 68 66 8 6 |

#include<bits/stdc++.h>

using namespace std;

bool cmp(string s,string s1){

if(s>s1) return true;

return false;

}

int main(){

ios\_base::sync\_with\_stdio(false); cin.tie(0);

int t; cin>>t;

while(t--)

{

int n; cin>>n;

priority\_queue<string, vector<string> , greater<string>> res;

vector<string> ans;

queue<string> pq;

pq.push("6");

pq.push("8");

while(pq.size()>0){

string s= pq.front();

pq.pop();

ans.push\_back(s);

if(s.size()>n) break;

pq.push(s+ "6");

pq.push(s +"8");

}

reverse(ans.begin(),ans.end());

for(auto &i: ans){

if(i.size()<=n) cout<<i<<" ";

}

cout<<endl;

}

}

GIÁ TRỊ NHỎ NHẤT CỦA XÂU

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08004/top)

Cho xâu ký tự S[] bao gồm các ký tự in hoa [A, B, …,Z]. Ta định nghĩa giá trị của xâu S[] là tổng bình phương số lần xuất hiện mỗi ký tự trong xâu. Ví dụ với xâu S[] = “AAABBCD” ta có F(S) = 32 + 22 + 12 + 12 = 15. Hãy tìm giá trị nhỏ nhất của xâu S[] sau khi loại bỏ K ký tự trong xâu.

**Input**:

* Dòng đầu tiên đưa vào số lượng test T (T≤100).
* Mỗi test được tổ chức thành 2 dòng. Dòng thứ nhất ghi lại số K. Dòng thứ 2 ghi lại xâu ký tự S[] có độ dài không vượt quá 10^6.

Output:

* Đưa ra giá trị nhỏ nhất của mỗi test theo từng dòng.

|  |  |
| --- | --- |
| Input | Output |
| 2  0  ABCC  1  ABCC | 6  3 |

#include<bits/stdc++.h>

typedef long long ll;

using namespace std;

long long solve(string str, ll k){

ll l = str.length();

if (k >= l)

return 0;

ll freq[26] = {0};

for (ll i = 0; i < l; i++)

freq[str[i]-'A']++;

priority\_queue<int> q;

for (ll i = 0; i < 26; i++)

q.push(freq[i]);

while (k--){

ll tmp = q.top();

q.pop();

tmp = tmp - 1;

q.push(tmp);

}

ll res = 0;

while (!q.empty()){

ll tmp = q.top();

q.pop();

res = res + tmp\*tmp;

}

return res;

}

int main(){

int t; cin >> t;

while (t--){

ll k; cin >> k;

string str; cin >> str;

cout << solve(str, k) << endl;

}

}

BIẾN ĐỔI S – T

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08009/top)

Cho hai số nguyên dương S và T (S, T<10000) và hai thao tác (a), (b) dưới đây:

**Thao tác (a)**: Trừ S đi 1  (S = S-1) ;

**Thao tác (b)**: Nhân S với 2 ( S = S\*2);

Hãy dịch chuyển S thành T sao cho số lần thực hiện các thao tác (a), (b) là ít nhất. Ví dụ với    S =2, T=5 thì số các bước ít nhất để dịch chuyển S thành T thông qua 4 thao tác sau:

**Thao tác (a)**: 2\*2 = 4;

**Thao tác (b)**: 4-1 = 3;

**Thao tác (a)**: 3\*2 = 6;

**Thao tác (b)**: 6-1 = 5;

**Input:**

* Dòng đầu tiên ghi lại số tự nhiên T là số lượng Test;
* T dòng kế tiếp mỗi dòng ghi lại một bộ Test. Mỗi test là một bộ đôi S và T.

**Output:** Đưa ra kết quả mỗi test theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  2 5  3 7  7 4 | 4  4  3 |

#include<bits/stdc++.h>

using namespace std;

struct node{

int val;

int level;

};

int solve(int x, int y){

queue<node> q;

node n = {x, 0};

q.push(n);

set<int> visited;

visited.insert(x);

while (!q.empty()){

node t = q.front();

q.pop();

if (t.val == y)

return t.level;

if (t.val\*2==y || t.val-1==y)

return t.level+1;

if (visited.find(t.val\*2)==visited.end() && t.val<y){

n.val = t.val\*2;

n.level = t.level+1;

q.push(n);

visited.insert(t.val\*2);

}

if (visited.find(t.val-1)==visited.end() && (t.val-1)>0){

n.val = t.val-1;

n.level = t.level+1;

q.push(n);

visited.insert(t.val-1);

}

}

}

int main(){

int t; cin >> t;

while (t--){

int x, y; cin >> x >> y;

cout << solve(x, y) <<endl;

}

}

BIẾN ĐỔI SỐ NGUYÊN TỐ

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA08020/top)

Cho cặp số S và T là các số nguyên tố có 4 chữ số (Ví dụ S = 1033, T = 8197 là các số nguyên tố có 4 chữ số). Hãy viết chương trình tìm cách dịch chuyển S thành T thỏa mãn đồng thời những điều kiện dưới đây:

1. Mỗi phép dịch chuyển chỉ được phép thay đổi một chữ số của số ở bước trước đó (ví dụ nếu S=1033 thì phép dịch chuyển S thành 1733 là hợp lệ);
2. Số nhận được cũng là một số nguyên tố có 4 chữ số (ví dụ nếu S=1033 thì phép dịch chuyển S thành 1833 là không hợp lệ, và S dịch chuyển thành 1733 là hợp lệ);
3. Số các bước dịch chuyển là ít nhất.

Ví dụ số các phép dịch chuyển ít nhất để S = 1033 thành  T = 8179 là 6 bao gồm các phép dịch chuyển như sau:

8179 ß 8779ß 3779ß 3739ß 3733ß 1733ß 1033.

**Input:**

* Dòng đầu tiên đưa vào số lượng test T (T≤100)
* Những dòng kế tiếp mỗi dòng đưa vào một test. Mỗi test là một bộ đôi S, T.

**Output:**

* Đưa ra kết quả mỗi test theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  1033 8179  1033 8779 | 6  5 |

#include<bits/stdc++.h>

using namespace std;

const long limit = 2e5+5;

long long x, y, t;

int d[limit] = {0};

vector<long long> a;

bool ok[limit];

int dem[limit];

vector<long long> ke[limit];

queue<long long> q;

void sieve(){

for (int i = 2; i <= 9999; i++){

if (d[i] == 0){

for (int j = i\*i; j <= 9999; j += i)

d[j] = 1;

}

}

for (int i = 1000; i <= 9999; i++)

if (d[i] == 0)

a.push\_back(i);

}

bool check(long long x, long long y){

int d = 0;

while (x && y){

if (x%10 != y %10) d++;

x /= 10; y /= 10;

}

return d == 1;

}

void init(){

for (int i = 0; i < a.size()-1; i++){

for (int j = i+1; j < a.size(); j++){

if (check(a[i], a[j])){

ke[a[i]].push\_back(a[j]);

ke[a[j]].push\_back(a[i]);

}

}

}

}

int BFS(long long u, long long d){

q.push(u);

dem[u] = 0;

while (q.size() > 0){

u = q.front();

q.pop();

ok[u] = true;

if (u == d) return dem[d];

for (int i = 0; i < ke[u].size(); i++){

long long v = ke[u][i];

if (ok[v]) continue;

ok[v] = true;

dem[v] = dem[u] + 1;

q.push(v);

}

}

}

int main(){

sieve();

init();

cin >> t;

while (t--){

for (int i = 0; i < limit; i++){

ok[i] = false;

dem[i] = 0;

}

while (q.size()) q.pop();

cin >> x >> y;

cout << BFS(x, y) << endl;

}

return 0;

}