# Mô hình thuật toán sinh kế tiếp

1. Biểu diễn tổng quát thuật toán

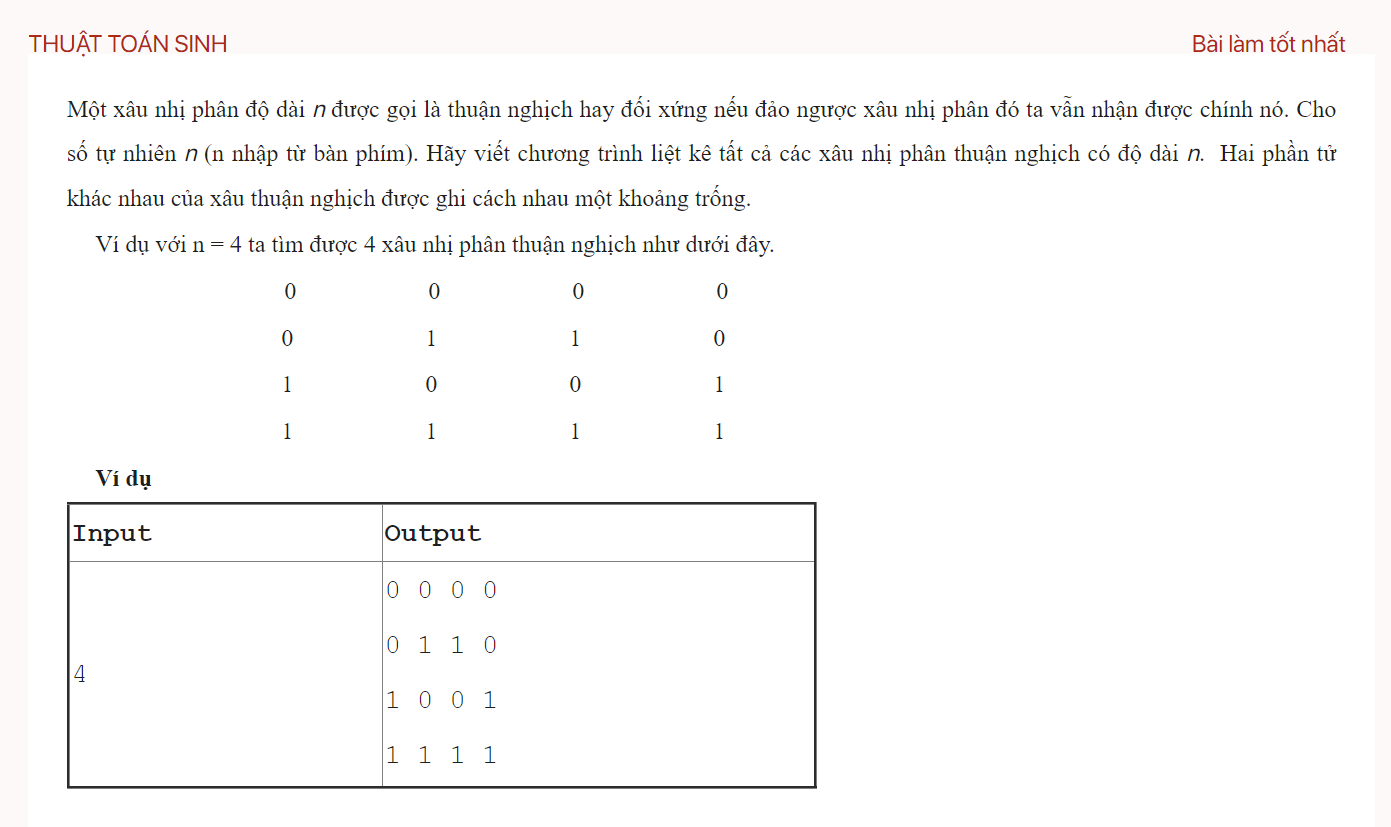
Thuật toán sinh tạo ra cấu hình kế tiếp dựa trên cấu hình hiện tại theo một quy luật nào đó cho đến khi cấu hình hiện tại là cấu hình cuối cùng.

VD sinh nhị phân:

* Quy luật: từ cấu hình hiện tại, tìm vị trí bit 0 đầu tiên từ phải sang, chuyển thành bit 1, các bit 1 phía sau chuyển thành 0.
* Lặp lại các bước trên cho đến khi cấu hình hiện tại là cấu hình cuối (trong xâu không còn bit 0 nào).

1. Ví dụ

[CTDL\_001 - THUẬT TOÁN SINH (ptit.edu.vn)](https://code.ptit.edu.vn/student/question/CTDL_001)



* Bài toán yêu cầu đưa ra tất cả các xâu nhị phân đối xứng có độ dài n. Nên có thể sử dụng thuật toán sinh kết hợp với việc kiểm tra điều kiện đối xứng để giải quyết bài toán.
* Code

#include <iostream>

using namespace std;

int a[100];

int n;

bool ok;

bool check() {

int l = 0, r = n - 1;

while(l < r) {

if(a[l] != a[r]) return false;

l += 1;

r -= 1;

}

return true;

}

void sinh() {

int i = n - 1;

while(i >= 0 && a[i] == 1) {

a[i] = 0;

i -= 1;

}

if(i < 0) ok = false;

else {

a[i] = 1;

}

}

int main() {

cin >> n;

for(int i = 0; i < n; i++) a[i] = 0;

ok = true;

while(ok) {

if(check()) {

for(int i = 0; i < n; i++) cout << a[i] << " ";

cout << endl;

}

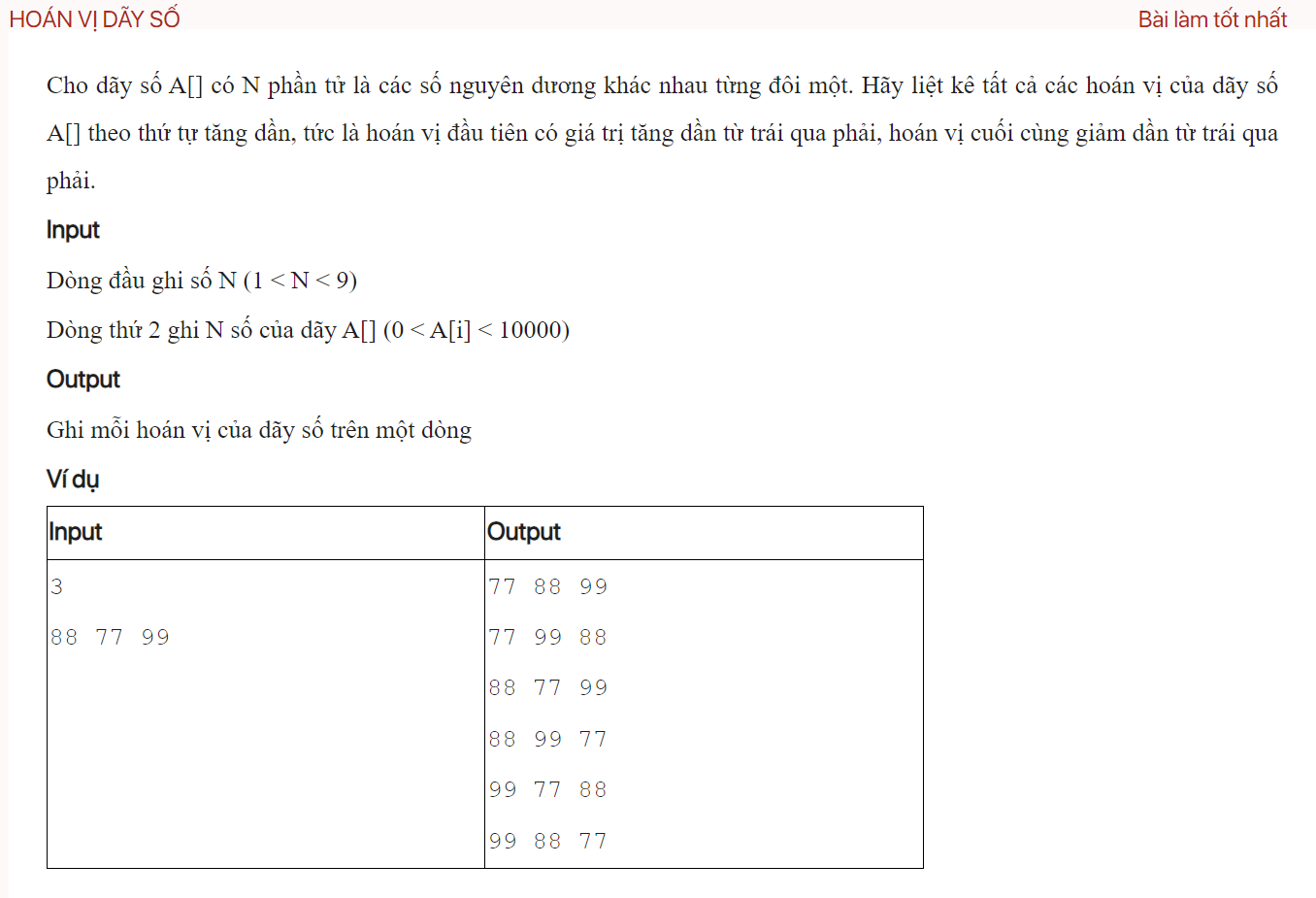
sinh();

}

return 0;

}

[DSA01027 - HOÁN VỊ DÃY SỐ (ptit.edu.vn)](https://code.ptit.edu.vn/student/question/DSA01027)



* Bài toán yêu cầu đưa ra tất cả hoán vị của các phần tử trong dãy. Ta có thể dùng sinh hoán vị để đưa ra tất cả hoán vị các vị trí trong dãy và ánh xạ vị trí để đưa ra giá trị của các phần tử.
* Code

#include <iostream>

#include <algorithm>

using namespace std;

int n;

int a[100];

int b[100];

bool ok;

void sinh() {

int i = n - 2;

while(i >= 0 && b[i] > b[i + 1]) {

i -= 1;

}

if(i < 0) ok = false;

else {

int j = n - 1;

while(j > i && b[j] < b[i]) {

j -= 1;

}

swap(b[i], b[j]);

i += 1;

j = n - 1;

while(i < j) {

swap(b[i], b[j]);

i += 1;

j -= 1;

}

}

}

int main() {

cin >> n;

for(int i = 0; i < n; i++) cin >> a[i];

for(int i = 0; i < n; i++) b[i] = i;

sort(a, a + n);

ok = true;

while(ok) {

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

cout << a[b[i]] << " ";

}

cout << endl;

sinh();

}

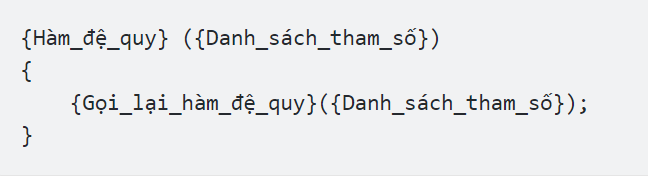
return 0;

}

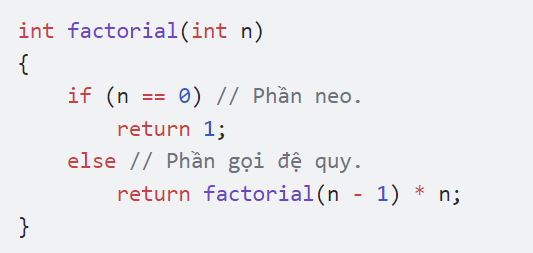
# Mô hình thuật toán đệ quy

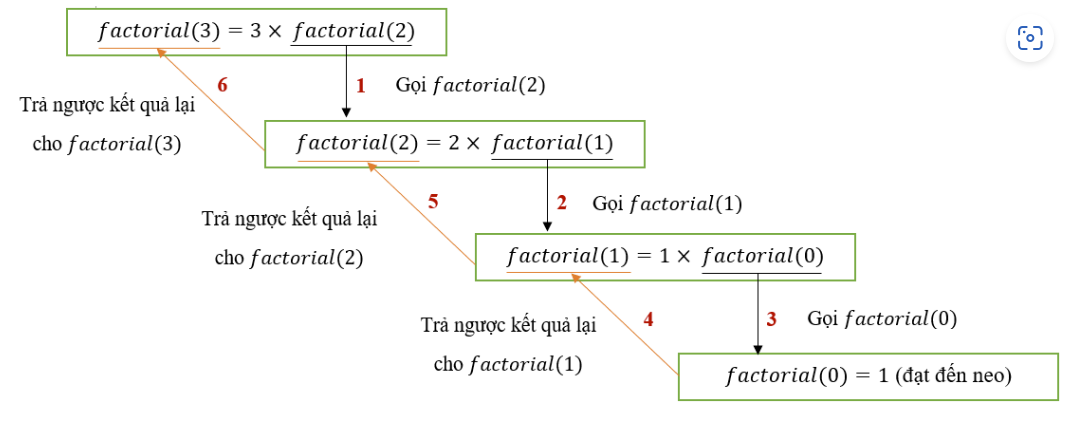
1. Biểu diễn tổng quát thuật toán

Mô hình thuật toán



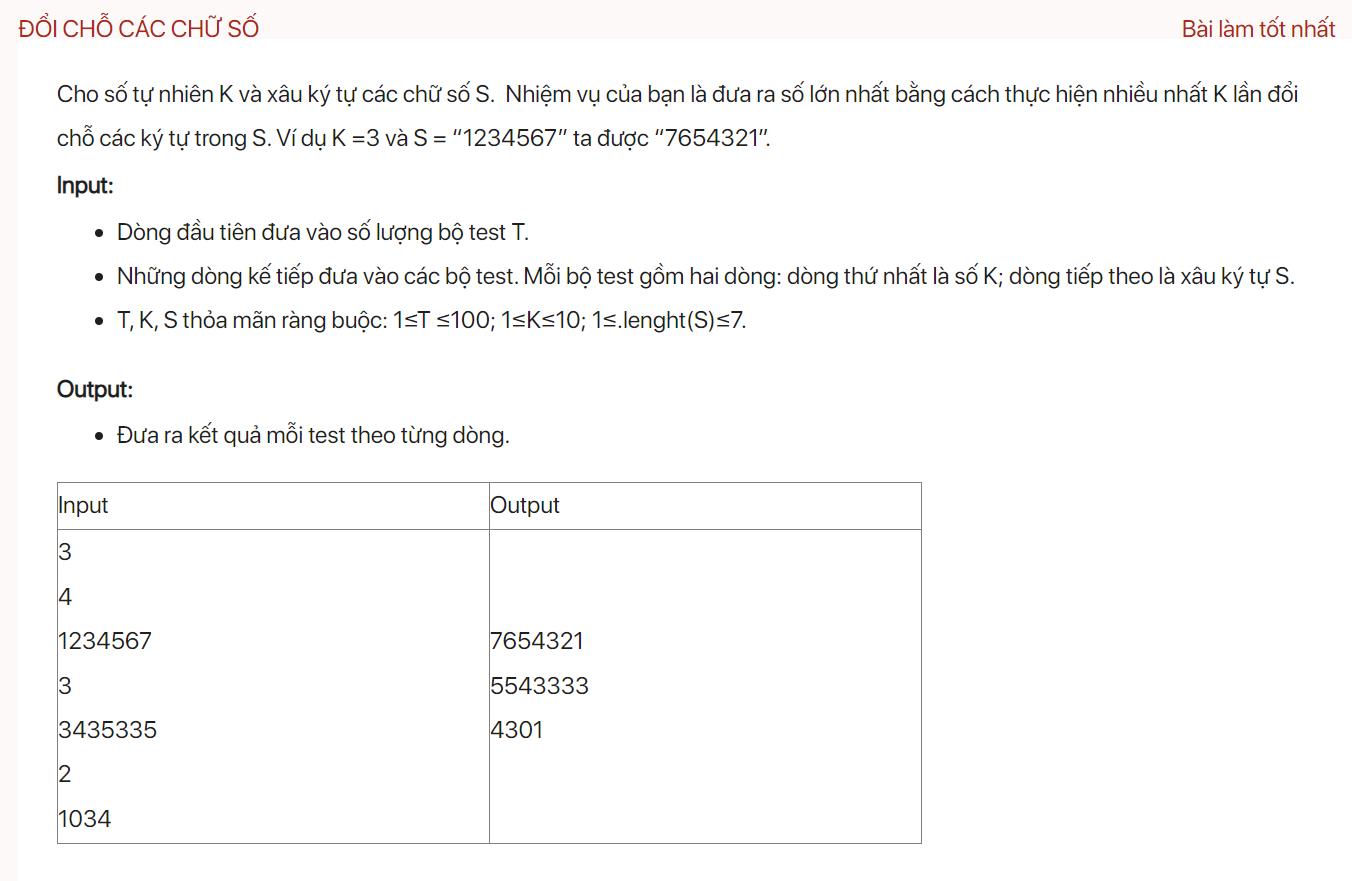
VD: tính giai thừa





1. Ví dụ

[DSA02007 - ĐỔI CHỖ CÁC CHỮ SỐ (ptit.edu.vn)](https://code.ptit.edu.vn/student/question/DSA02007)



* Ý tưởng tiếp cận bằng đệ quy: tại mỗi vị trí i ta sẽ cố gắng tìm một vị trí j nào đó nằm bên phải i, xa i nhất, có giá trị a[j] lớn nhất và lớn hơn a[i] để đổi chỗ. Nếu đã tìm được vị trí j tương ứng cho vị trị i đang xét ta sẽ đổi chỗ a[i], a[j] và gọi đệ quy để xử lý vị trí i + 1 và số lần đổi chỗ giảm đi 1. Nếu không tìm được vị trí j thỏa mãn ta sẽ gọi đệ quy để xử lý vị trí i + 1 với k giữ nguyên. Khi đã thực hiện đổi chỗ k lần hoặc i đã là vị trí cuối cùng thì trả về xâu s là xâu lớn nhất.
* Code

#include <iostream>

using namespace std;

string solve(string s, int i, int k) {

if(k == 0 || i >= s.length()) {

return s;

}

int idx = i;

for(int j = i + 1; j < s.length(); j++) {

if(s[j] >= s[idx]) {

idx = j;

}

}

if(s[idx] != s[i]) {

swap(s[i], s[idx]);

return solve(s, i + 1, k - 1);

}

return solve(s, i + 1, k);

}

int main() {

int t;

cin >> t;

while(t--) {

int k;

cin >> k;

string s;

cin >> s;

cout << solve(s, 0, k) << endl;

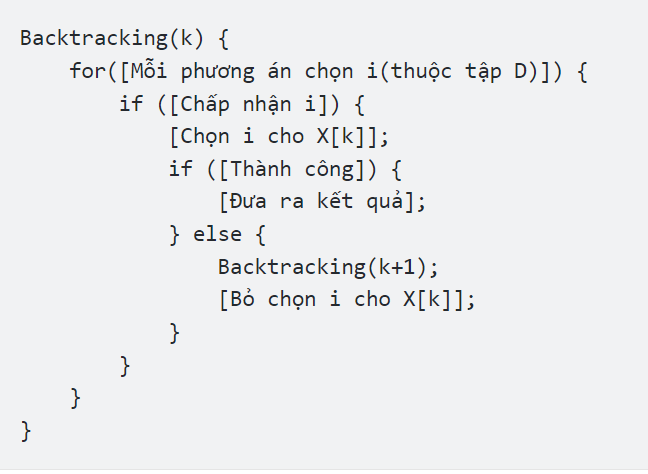
}

return 0;

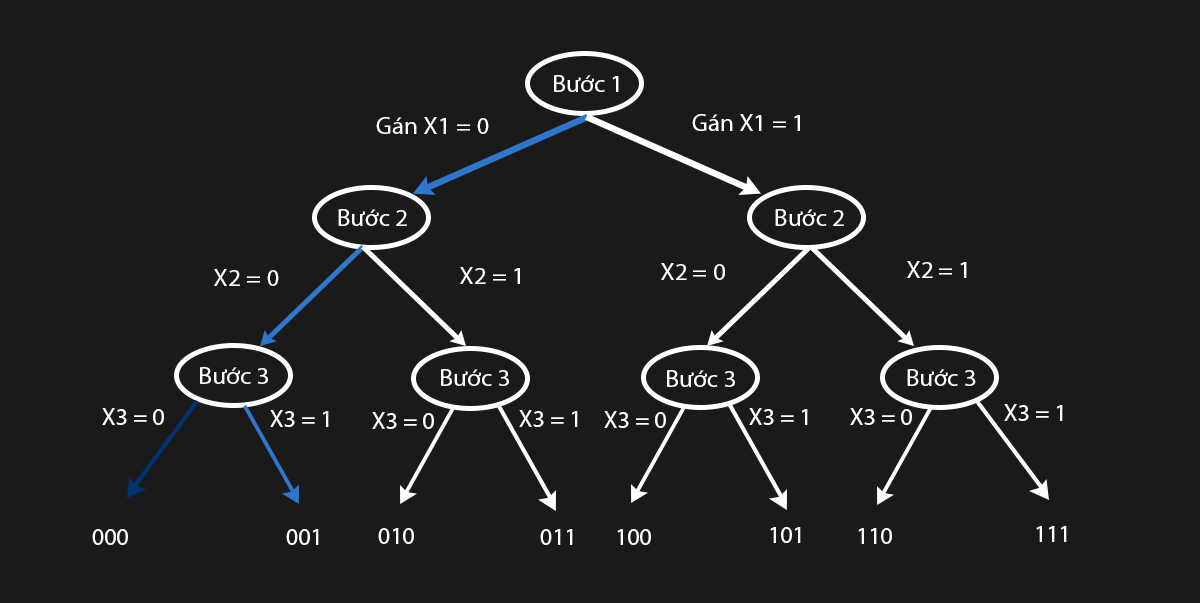
}

# Mô hình thuật toán quay lui

1. Biểu diễn tổng quát thuật toán

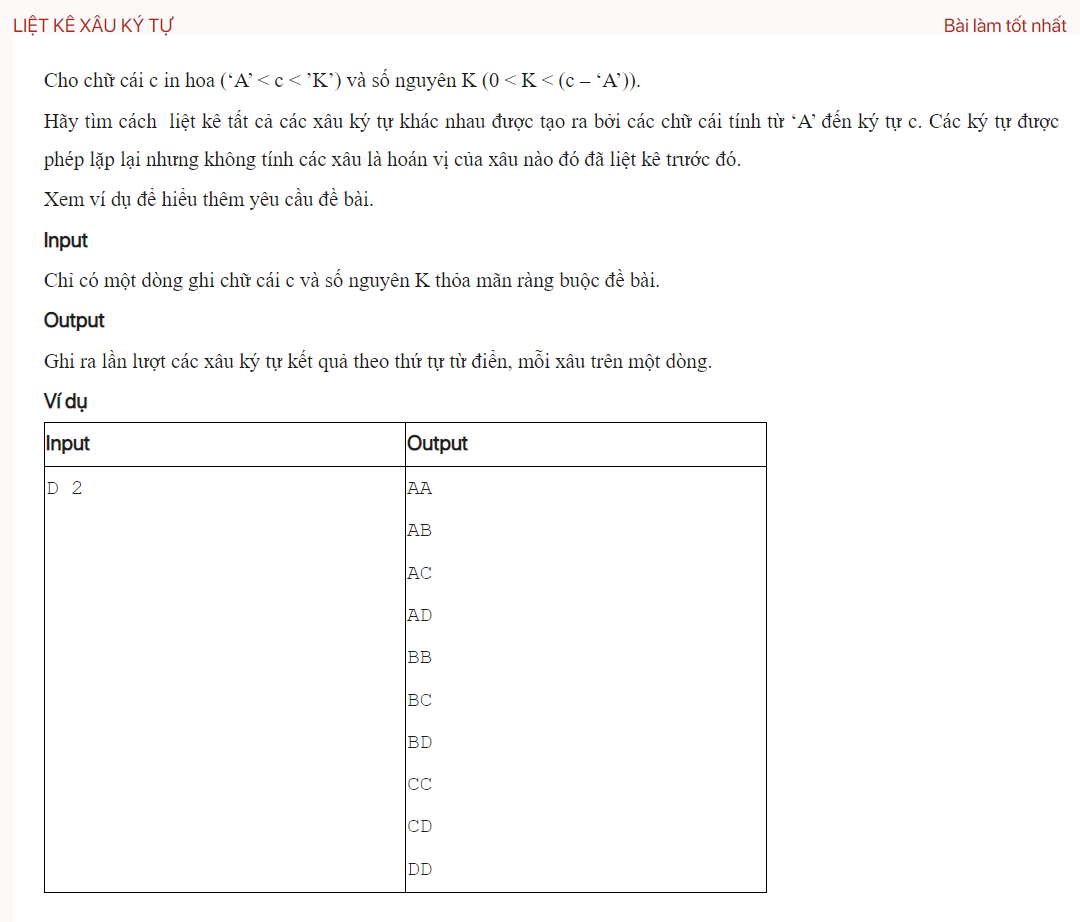


* Quay lui nhị phân



1. Ví dụ

[DSA02030 - LIỆT KÊ XÂU KÝ TỰ (ptit.edu.vn)](https://code.ptit.edu.vn/student/question/DSA02030)



* Tại mỗi vị trí i trong xâu đều có giá trị trong đoạn từ ‘A’ đến c, vì thế ta có thể sử dụng thuật toán quay lui để giải quyết bài này.
* Code

#include <iostream>

using namespace std;

int a[100];

int n, k;

void quayLui(int i) {

for(int j = a[i - 1]; j <= n; j++) {

a[i] = j;

if(i == k) {

for(int idx = 1; idx <= k; idx++) {

cout << (char) ('A' + a[idx]);

}

cout << endl;

}

else quayLui(i + 1);

}

}

int main() {

char c;

cin >> c >> k;

a[0] = 0;

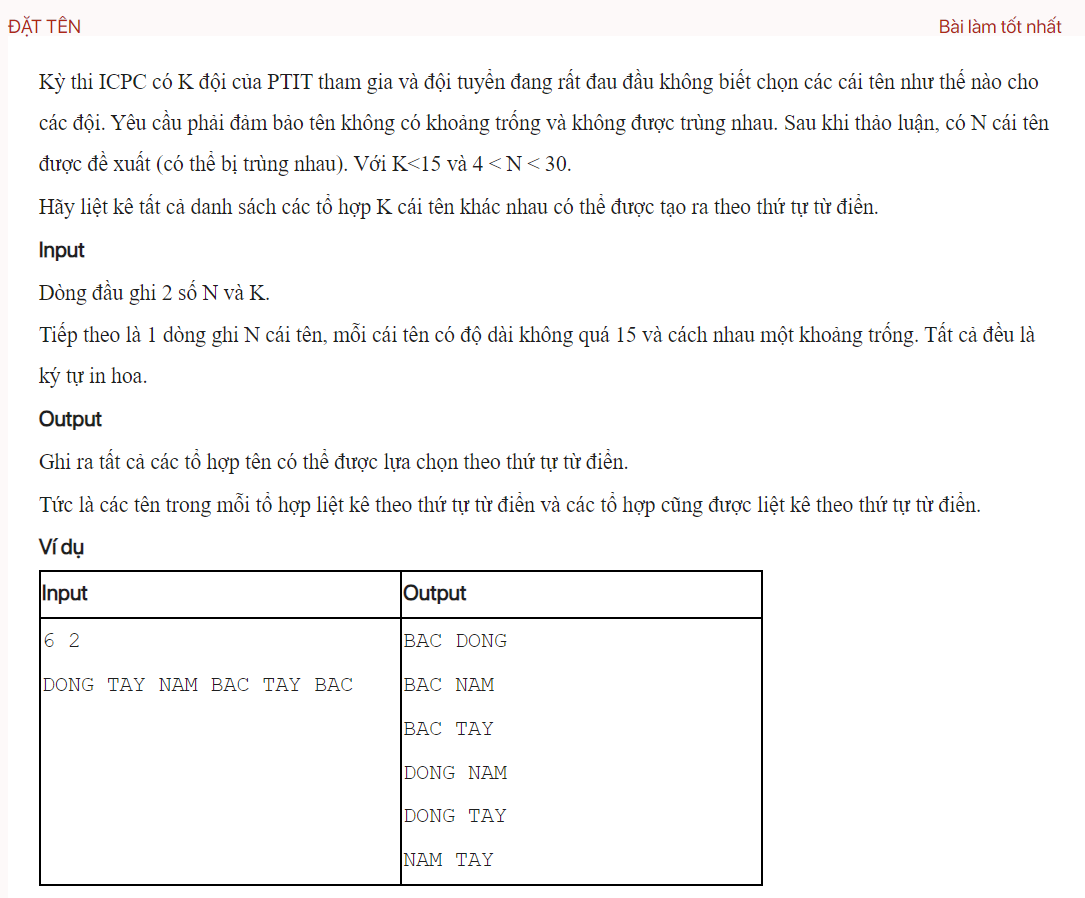
n = (int) (c - 'A');

quayLui(1);

return 0;

}

[DSA02023 - ĐẶT TÊN (ptit.edu.vn)](https://code.ptit.edu.vn/student/question/DSA02023)



* Bài yêu cầu đưa ra tất cả các cách chọn k tên trong số các tên đã cho. Để giải quyết bài này có thể sử dụng quay lui sinh ra tổ hợp chập k của n với n là số lượng các tên phân biệt.
* Code

#include <iostream>

#include <set>

using namespace std;

int a[100];

int n, k;

set<string> s;

void quayLui(int i) {

for(int j = a[i - 1] + 1; j < n - k + i; j++) {

a[i] = j;

if(i == k) {

for(int idx = 1; idx <= k; idx++) {

cout << \*(next(s.begin(), a[idx])) << " ";

}

cout << endl;

}

else quayLui(i + 1);

}

}

int main() {

cin >> n >> k;

a[0] = -1;

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

string x;

cin >> x;

s.insert(x);

}

n = s.size();

quayLui(1);

return 0;

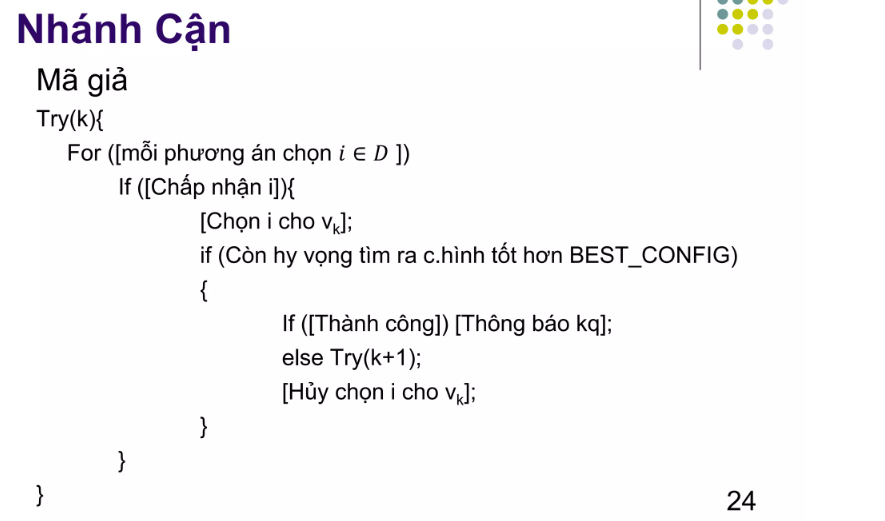
}

# Mô hình thuật toán nhánh cận

1. Biểu diễn tổng quát thuật toán

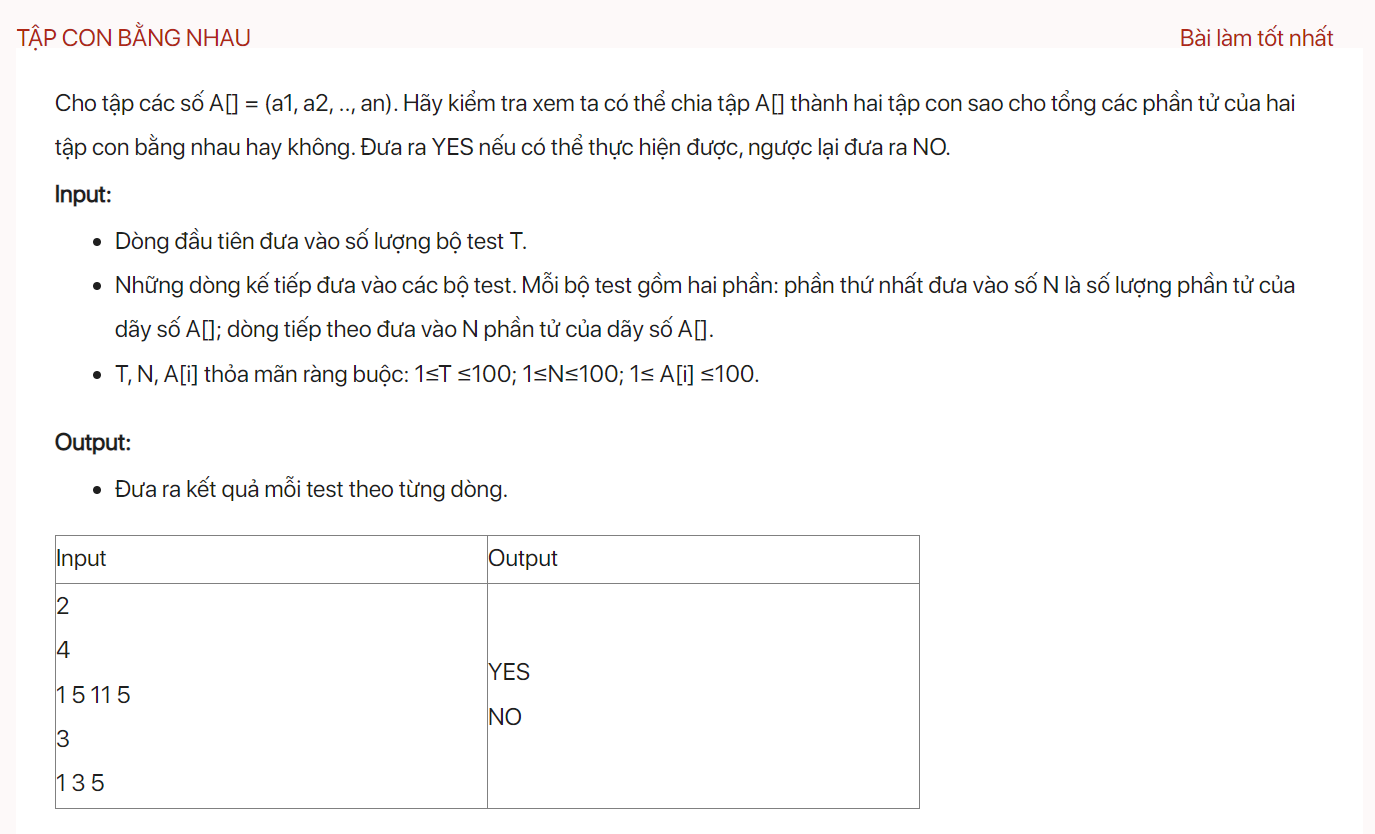
Nhánh cận là một kỹ thuật đánh giá việc tiếp tục đào sâu có tạo ra cấu hình tốt hơn cấu hình tốt nhất mà ta đang lưu trữ hay không.

Nhờ có nhánh cận mà ta có thể đưa ra quyết định quay lui sớm hơn thuật toán quay lui cổ điển.



1. Ví dụ

[DSA05009 - TẬP CON BẰNG NHAU (ptit.edu.vn)](https://code.ptit.edu.vn/student/question/DSA05009)



* Nếu mảng A có thể chia thành 2 tập có tổng bằng nhau thì sẽ tồn tại 1 cách chọn ra một tập B là con của tập A và có tổng bằng ½ tổng của mảng A. Tại mỗi vị trí i trong mảng A ta có thể thêm A[i] vào B hoặc không. A[i] chỉ có thể thêm vào B nếu tổng B hiện tại + A[i] <= ½ tổng của A. thuật toán dừng lại khi tổng B bằng ½ tổng A hoặc khi đã duyệt tất cả các trường hợp nhưng không tìm ra kết quả.
* Code

#include <iostream>

using namespace std;

int a[100];

int n;

int S;

bool kq = false;

void quayLui(int i, int sum) {

if(kq) return;

if(sum \* 2 == S) {

kq = true;

return;

}

if(i < n) {

if((a[i] + sum) \* 2 <= S) {

quayLui(i + 1, sum + a[i]);

}

quayLui(i + 1, sum);

}

}

int main() {

ios\_base::sync\_with\_stdio(0);

cin.tie(0);

int t;

cin >> t;

while(t--) {

cin >> n;

S = 0;

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

cin >> a[i];

S += a[i];

}

kq = false;

if(S % 2 == 0) quayLui(0, 0);

if(kq) cout << "YES" << endl;

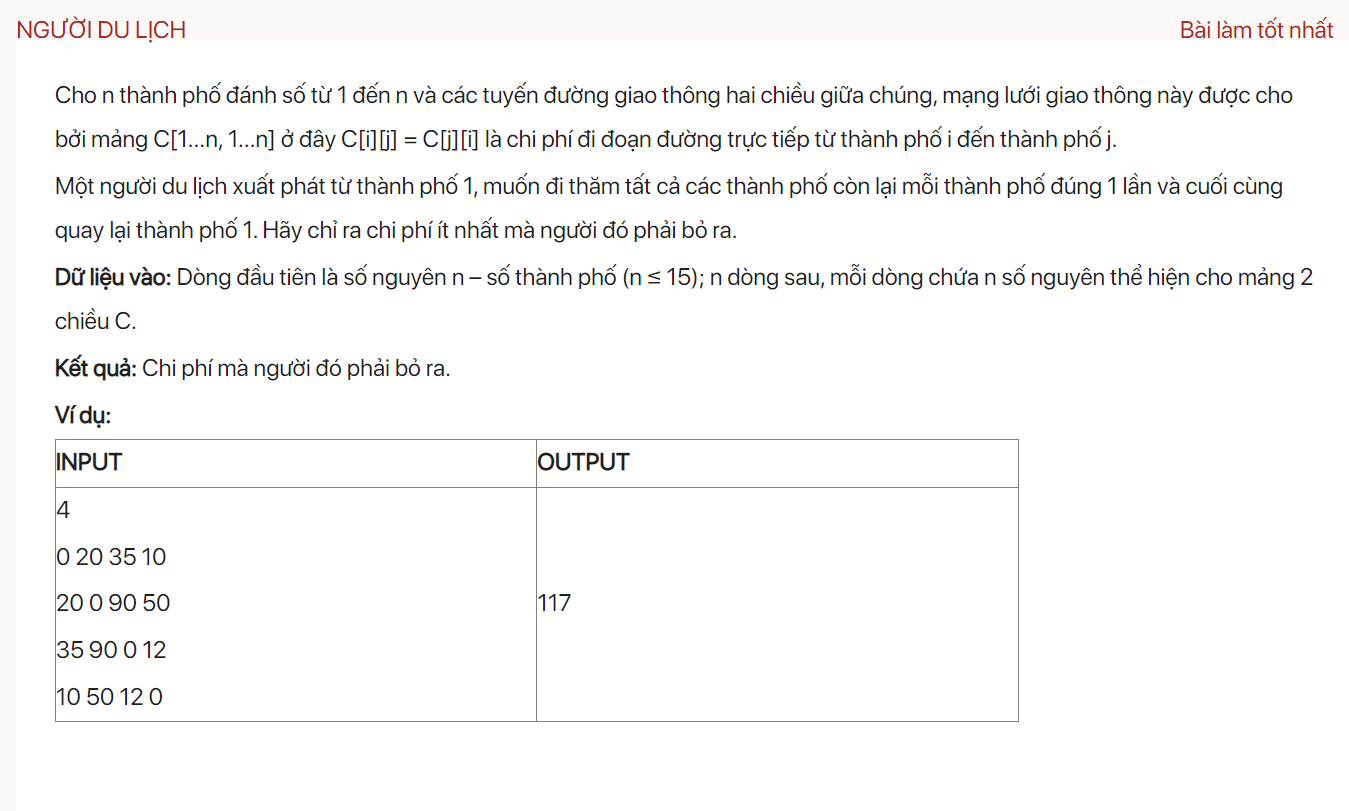
else cout << "NO" << endl;

}

return 0;

}

[DSA02027 - NGƯỜI DU LỊCH (ptit.edu.vn)](https://code.ptit.edu.vn/student/question/DSA02027)



* Bài toán yêu cầu tìm ra lộ trình đi từ 1 qua tất cả các thành phố còn lại sau đó quay lại thành phố 1, đây chín là quay lui sinh ra tất cả các hoán vị từ 2 đến <= n vì vị trí số 0 luôn cố định là thành phố 1. Để giải bài này ta sẽ kết hợp quay lui sinh hoán vị với nhánh cận: chỉ đi vào các trường hợp có kết quả tốt hơn kết quả hiện tại.
* Code

#include <iostream>

using namespace std;

int a[20][20];

int n;

bool used[20];

int kq;

void quayLui(int i, int cur, int sum) {

for(int j = 1; j < n; j++) {

if(!used[j] && sum + a[cur][j] < kq) {

used[j] = true;

sum += a[cur][j];

if(i == n - 2) {

kq = min(kq, sum + a[j][0]);

}

else quayLui(i + 1, j, sum);

sum -= a[cur][j];

used[j] = false;

}

}

}

int main() {

cin >> n;

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

used[i] = false;

for(int j = 0; j < n; j++) {

cin >> a[i][j];

}

}

kq = 1e9;

quayLui(0, 0, 0);

cout << kq;

return 0;

}