MÔ HÌNH KỸ THUẬT QUAY LUI

void quaylui(i). // Xây dựng thành phần số thứ i

{

for(xi khả năng các giá trị có thể của xi)

{

<ghi nhận thành phần số thứ i>

if(xâu dựng xong)

<Kiểm tra đánh giá và chọn nghiệm>

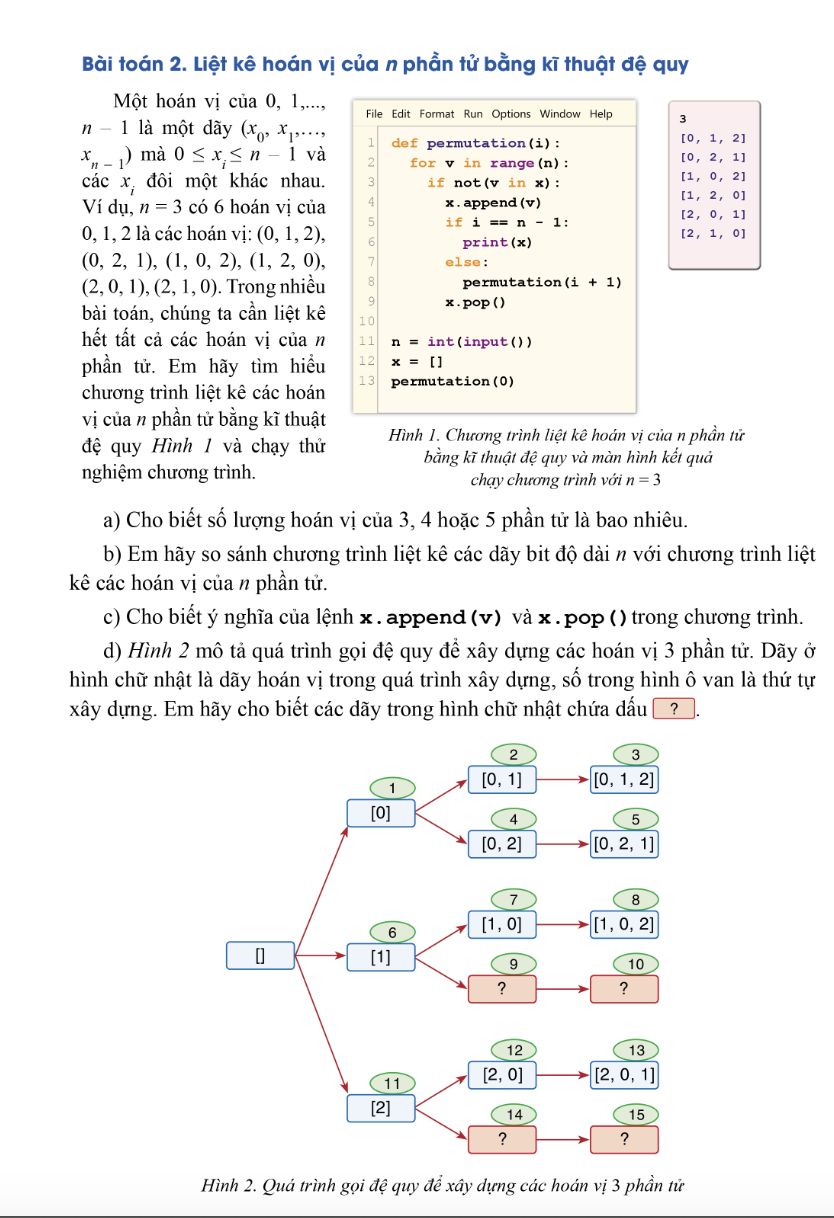
else

quaylui(i+1)

<Bỏ ghi nhận thành phần số i>

}

BÀI 1.

#include<bits/stdc++.h>

using namespace std;

void lietke(int i, int n, vector<int>& x) {

for (int v = 0; v < 2; ++v) {

x.push\_back(v); // Thêm bit vào dãy hiện tại

if (i == n - 1) {

// Khi đạt độ dài n, in ra dãy bit

for (int bit : x) {

cout << bit;

}

cout << endl;

} else {

// Gọi đệ quy để tiếp tục xây dựng dãy bit

lietke(i + 1, n, x);

}

x.pop\_back(); // Loại bỏ bit vừa thêm để thử các kết hợp khác

}

}

int main() {

int n; // Độ dài của dãy bit

// Nhập độ dài của dãy bit

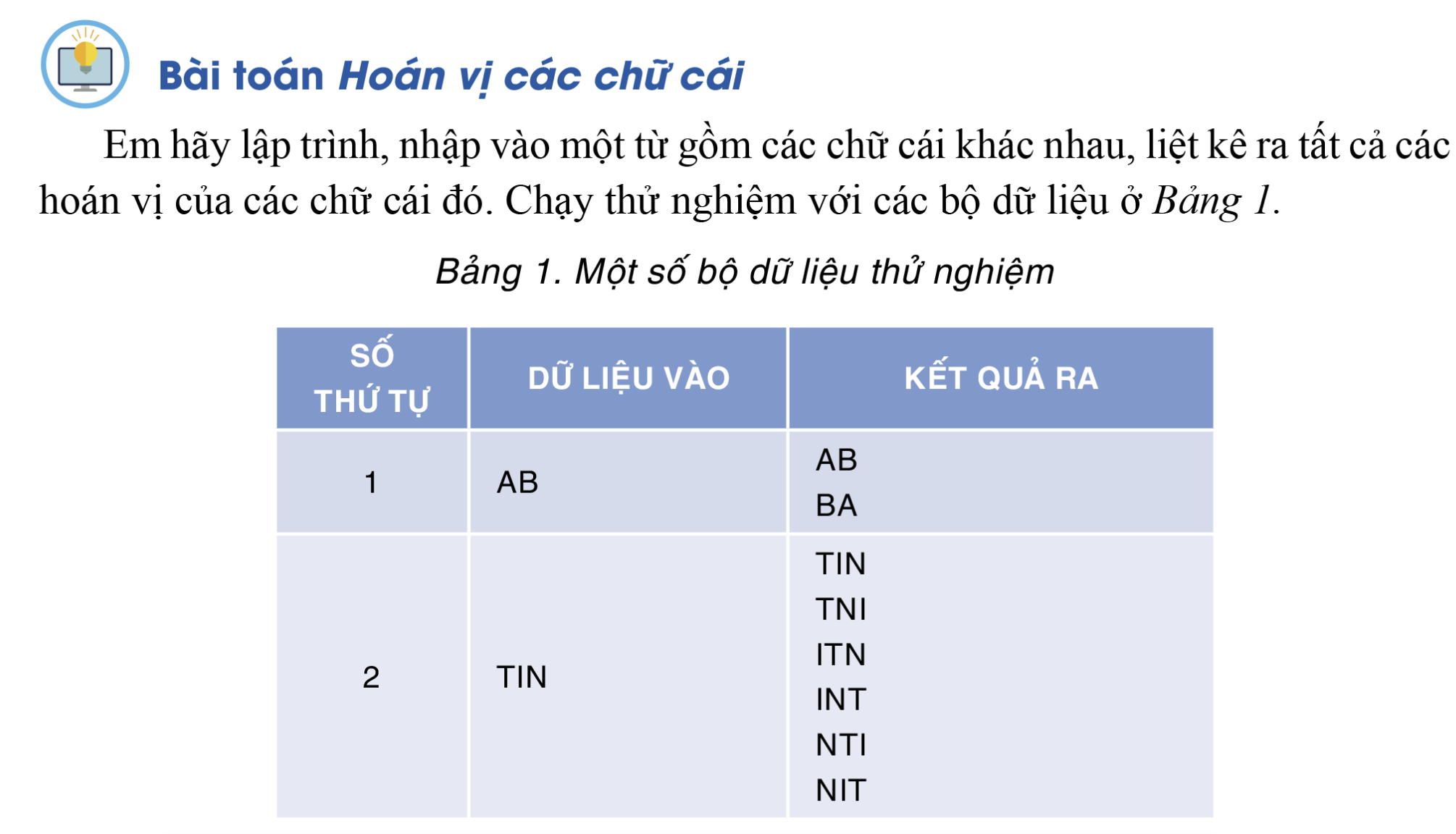
cin >> n;

vector<int> x; // Dãy bit hiện tại

lietke(0, n, x); // Gọi hàm đệ quy bắt đầu từ độ sâu 0

return 0;

}



BÀI 2

#include<bits/stdc++.h>

using namespace std;

void hoanvi(string &str, int l, int r) {

// Nếu tất cả các ký tự đã được hoán đổi

if (l == r) {

cout << str << endl;

} else {

for (int i = l; i <= r; i++) {

// Hoán đổi ký tự l và i

swap(str[l], str[i]);

// Đệ quy với phần còn lại của chuỗi

hoanvi(str, l + 1, r);

// Khôi phục trạng thái chuỗi ban đầu

swap(str[l], str[i]);

}

}

}

int main() {

string s;

cin >> s;

// Sắp xếp từ để đảm bảo các hoán vị được in theo thứ tự từ điển

sort(s.begin(), s.end());

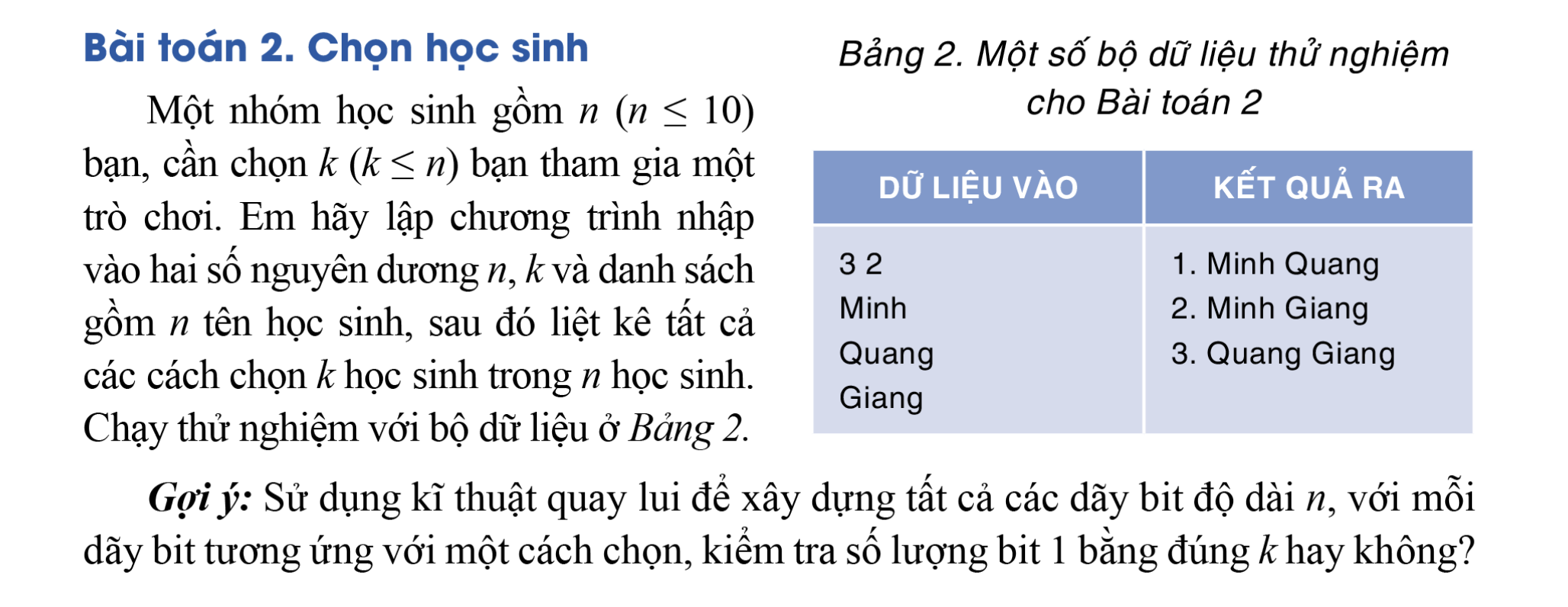
// Gọi hàm hoanvi để tạo và in tất cả các hoán vị

hoanvi(s, 0, s.length() - 1);

return 0;

}

Bài 3



#include<bits/stdc++.h>

using namespace std;

// Hàm đệ quy sinh tất cả các tổ hợp của k phần tử từ n phần tử

void chon(const vector<string>&a, vector<string>&ans, int start, int k)

{

// Nếu đã chọn đủ k học sinh, in kết quả

if (k == 0) {

for (const string& student : ans) {

cout <<student << " ";

}

cout << endl;

return;

}

// Tạo tổ hợp từ phần tử hiện tại

for (int i = start; i <= a.size() - k; ++i) {

ans.push\_back(a[i]);

// Tiếp tục chọn phần tử tiếp theo

chon(a, ans, i + 1, k - 1);

// Quay lại trạng thái trước đó

ans.pop\_back();

}

}

int main() {

int n, k;

vector<string> ans;

// Nhập số lượng học sinh (n) và số học sinh cần chọn (k)

cin >> n>>k;

vector<string> a(n);

// Nhập tên học sinh

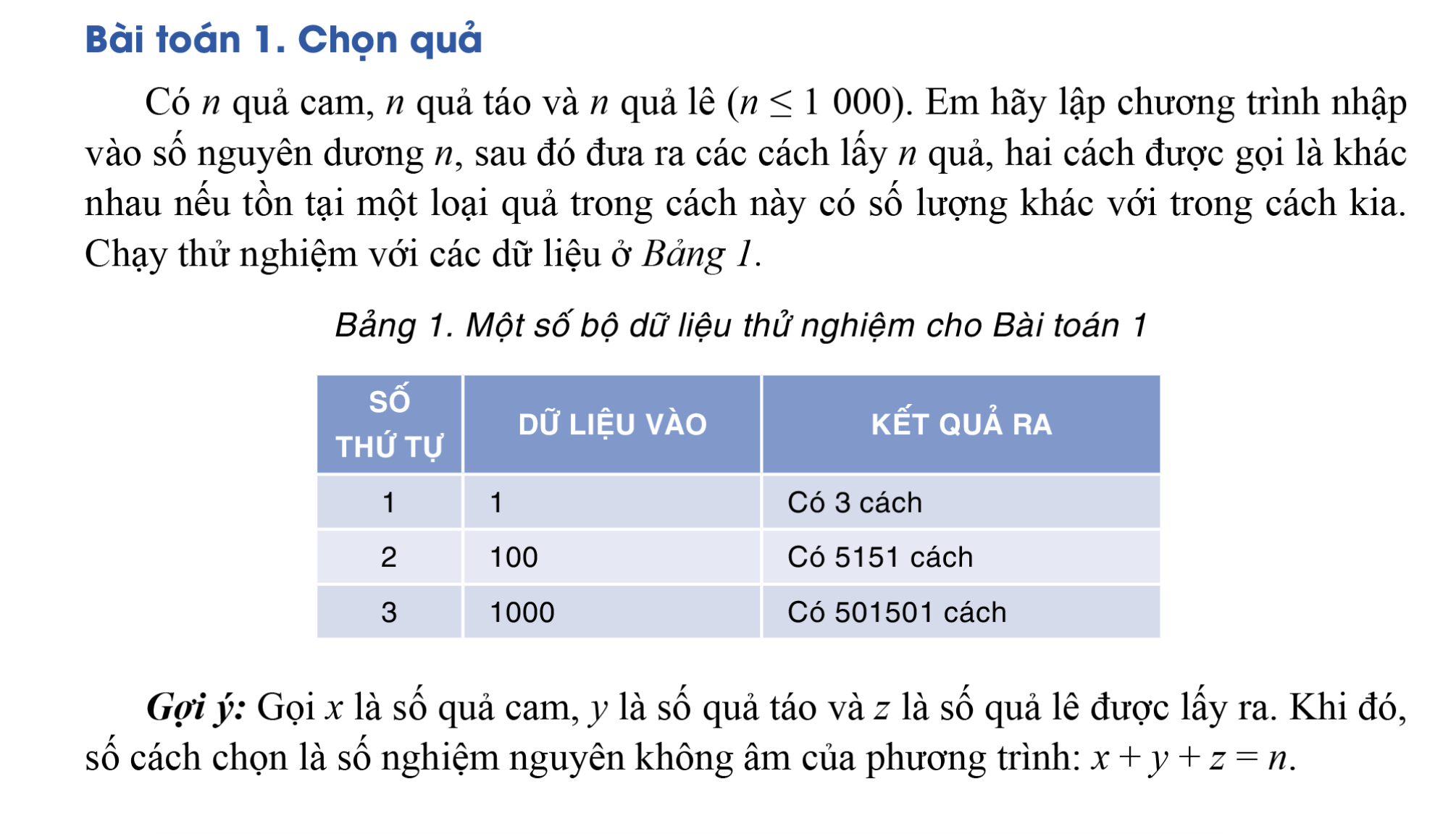
for (int i = 0; i < n; ++i) cin >> a[i];

chon(a, ans, 0, k);

return 0;

}

Bài 4



**Với bài toán x + y + z = n, chúng ta cần đếm số cách phân phối n quả thành 3 loại quả (x, y, z). Đây là bài toán phân phối với điều kiện.**

**Tính Số Cách Chọn:**

Có thể áp dụng công thức **số cách phân phối n đối tượng vào k nhóm** với điều kiện rằng số lượng các đối tượng trong từng nhóm có thể khác nhau. Công thức này là **số cách phân phối n đối tượng vào k nhóm (hay gọi là số tổ hợp với lặp lại)**.

Công thức tính số cách phân phối n đối tượng vào k nhóm là:

**C(n+k−1,k−1)**

Trong bài toán của chúng ta, n là số quả cần chọn và k là số loại quả (3 loại: cam, táo, lê).

 **Sử dụng phép biến hình**: Ta có thể hình dung mỗi phân phối tương đương với việc giải một bài toán về số cách sắp xếp nnn đối tượng và k−1k-1k−1 dấu phân cách (để chia nhóm). Ví dụ, nếu bạn có nnn đối tượng và bạn cần phân phối chúng vào kkk nhóm, bạn có thể tưởng tượng việc đặt k−1k-1k−1 dấu phân cách vào giữa các đối tượng để phân chia chúng thành kkk nhóm.

 **Số cách sắp xếp**: Số cách phân phối có thể được tính bằng cách chọn vị trí cho k−1k-1k−1 dấu phân cách trong số n+k−1n + k - 1n+k−1 vị trí (tổng số đối tượng và dấu phân cách). Công thức tổ hợp (n+k−1k−1)\binom{n+k-1}{k-1}(k−1n+k−1​) chính là số cách chọn k−1k-1k−1 vị trí từ n+k−1n+k-1n+k−1 vị trí.

#include <iostream>

using namespace std;

// Hàm tính tổ hợp C(n, k) mà không cần tính giai thừa

unsigned long long ckn(int n, int k) {

if (k > n) return 0;

if (k > n - k) k = n - k; // Vì C(n, k) = C(n, n - k)

unsigned long long result = 1;

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

result = result \* (n - i) / (i + 1);

}

return result;

}

int main() {

int n;

cin >> n;

// Kiểm tra tính hợp lệ của n

if (n <= 0) {

cout << "n <0" << endl;

return 1;

}

// Tính số cách chọn

// C(n + 2, 2)

unsigned long long result = ckn(n + 2, 2);

// In số cách chọn

cout << result << endl;

return 0;

}

// mowi

#include <iostream>

using namespace std;

int main() {

int n;

cin >> n;

// Kiểm tra tính hợp lệ của n

if (n <= 0)

cout << "n <0" << endl;

else

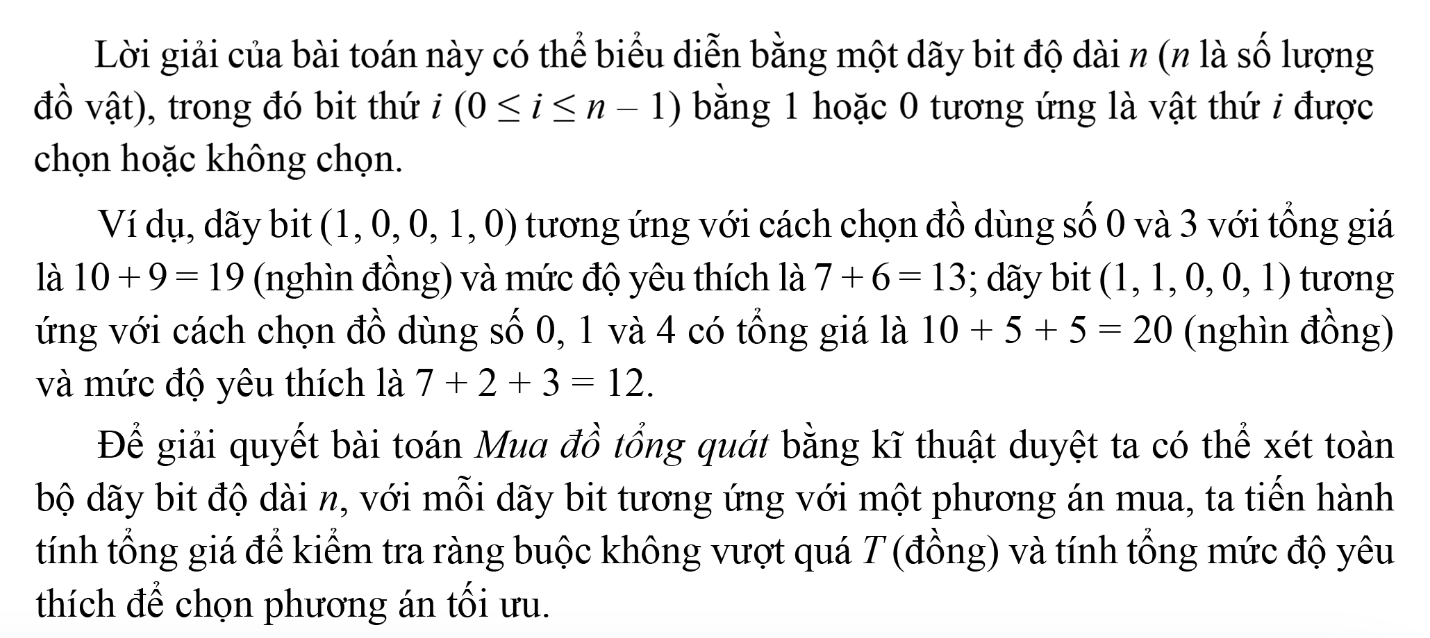
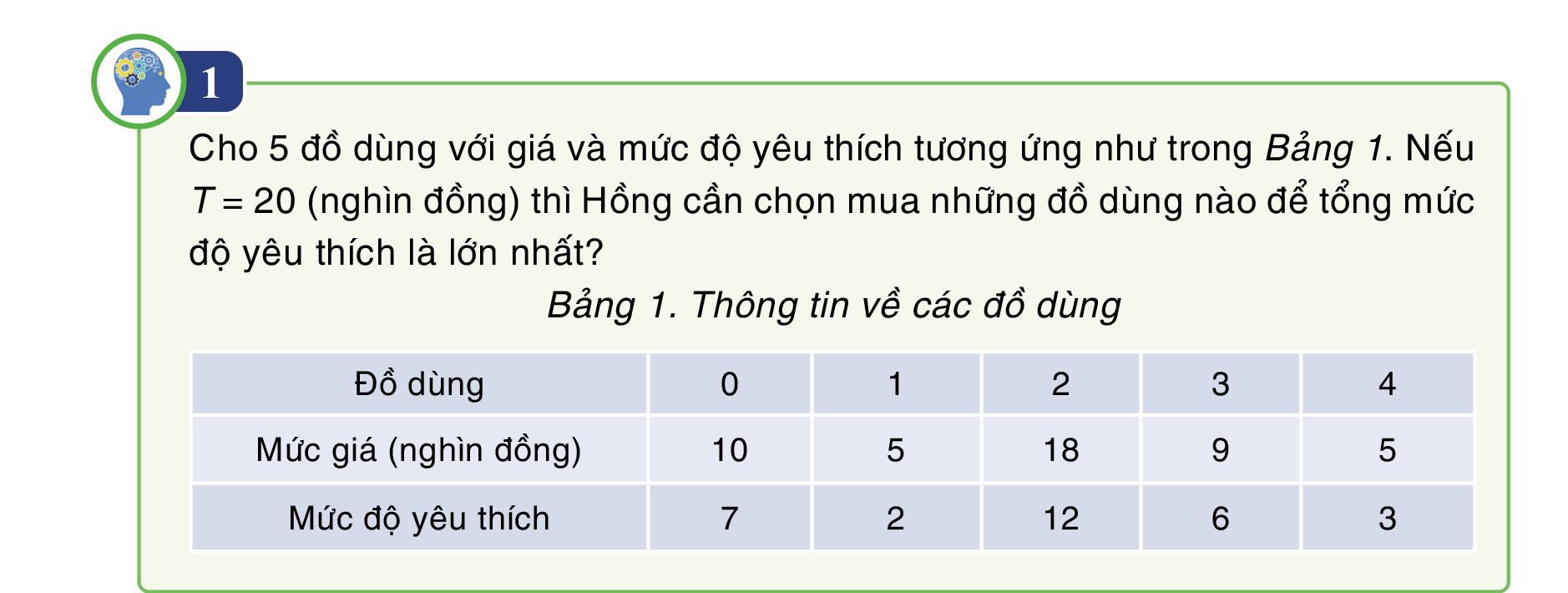
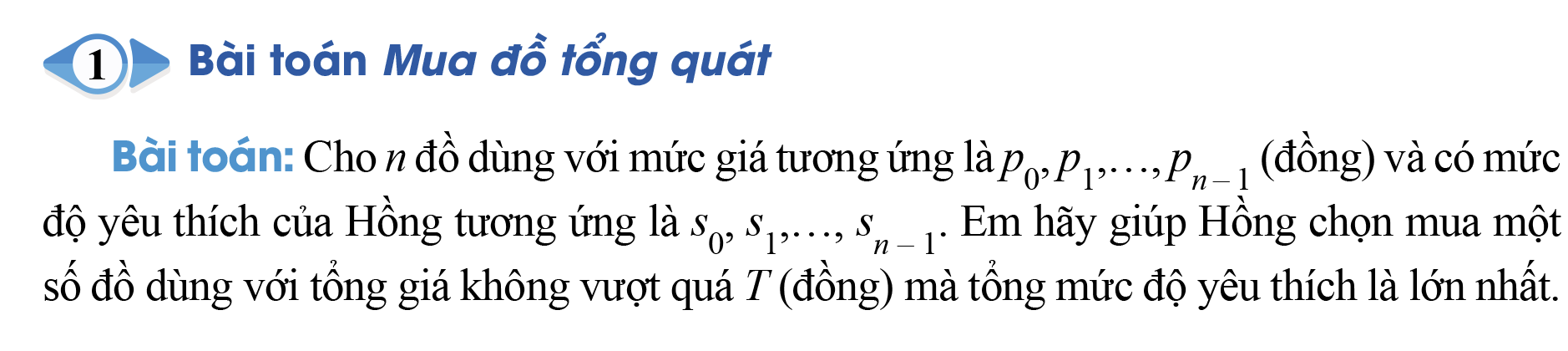
// In số cách chọn

cout << (n+2)\*(n+1)/2 << endl;

return 0;

}

BÀI 5

 using namespace std;

// Hàm đệ quy thực hiện quay lui

void quaylui(int index, int Cost, int Love, vector<int> a,vector<int>& ans, int& maxLove, int T, const vector<int>& P, const vector<int>& S) {

// Nếu vượt quá ngân sách, quay lại

if (Cost > T) return;

if (Love > maxLove) {

maxLove = Love;

ans = a;

}

// Duyệt các đồ dùng tiếp theo

for (int i = index; i < P.size(); ++i) {

// Thêm đồ dùng hiện tại vào danh sách đồ dùng được chọn

a.push\_back(i);

quaylui(i + 1, Cost + P[i], Love + S[i], a, ans, maxLove, T, P, S);

// Loại bỏ đồ dùng hiện tại khỏi danh sách để thử các kết hợp khác

a.pop\_back();

}

}

int main() {

int n; // Số lượng đồ dùng

int T; // Ngân sách tối đa

// Nhập dữ liệu

cin >> n>>T;

vector<int> P(n); // Mảng giá

vector<int> S(n); // Mảng mức độ yêu thích

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

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

vector<int> ans; // Danh sách các đồ dùng được chọn cho kết quả tối ưu

vector<int> a; // Danh sách các đồ dùng hiện tại trong quá trình quay lui

int maxLove = 0; // Mức độ yêu thích cao nhất

// Gọi hàm quay lui bắt đầu từ chỉ số 0

quaylui(0, 0, 0, a, ans, maxLove, T, P, S);

cout << maxLove << endl;

for (int index : ans) cout << index << " ";

cout << endl;

return 0;

}