Bài thực hành 3: Đệ quy và khử đệ quy để giải quyết một số bài toán

Phần 1. Thực hành về đệ quy

1.1 Đệ quy - quay lui

Bài tập 1: Tính dãy Lucas

Dãy Lucas được định nghĩa bởi $L_n = L_{n-1} + L_{n-2}$ và bắt đầu bởi $L_0 = 2$, $L_1 = 1$. Viết hàm tính số Lucas thứ n.

In [ ]:

```
int lucas(int n) {

    /*****************
    # YOUR CODE HERE #
    *****************/
}
```

Bài tập 2: Quân mã đi tuần

Trên bàn cờ vua kích thước $n \times n$ có một quân mã đang ở ô (1, 1). Hãy đưa ra một dãy các di chuyển của mã sao cho mỗi ô trên bàn cờ đều được đi qua đúng 1 lần (ô (1, 1) được xem là đã đi qua)

In [ ]:

```
#include <iostream>
using namespace std;

int n;
int X[100], Y[100]; //# Lưu tọa độ các bước di chuyển của quân mã
int mark[100][100]; //# Đánh dấu vị trí các ô mà quân mã đã di chuyển qua

//# Mảng hx, hy mô tả 8 vị trí quân mã có thể di chuyển kể từ vị trí hiện tại
const int hx[] = {1, 1, 2, 2, -1, -1, -2, -2};
const int hy[] = {2, -2, 1, -1, 2, -2, 1, -1};

//# In ra dãy các di chuyển tìm được
void print_sol(){
    for (int j = 1; j <= n * n; ++j)
        printf("(%d %d)\n", X[j], Y[j]);
    exit(0);
}

//# Thuật toán quay lui
void TRY(int k){
    for(int i = 0; i < 8; i++){
        int xx = X[k-1] + hx[i];
        int yy = Y[k-1] + hy[i];
        /*****************
        # YOUR CODE HERE #
        *****************/
    }
}

int main(){
    cin >> n;
    mark[1][1] = 1;
    X[1] = Y[1] = 1;
    TRY(2);
```

```
    return 0;
}
```

1.2 Kỹ thuật nhánh cận

Bài tập 3: Bài toán người du lịch

Một người xuất phát tại thành phố 1, muốn đi thăm tất cả các thành phố khác, mỗi thành phố đúng 1 lần và quay về 1. Chi phí để đi từ thành phố i sang thành phố j là $c_{i,j}$. Hãy tìm tổng chi phí nhỏ nhất có thể

Dữ liệu vào:

Dòng 1: Chứa số nguyên n ($1 \le n \le 16$)

n dòng tiếp theo: Chứa ma trận c ($0 \le c_{i,j} \le 1000000$)

Kết quả:

Ghi tổng chi phí nhỏ nhất có thể

Ví dụ:

Dữ liệu mẫu: 4

0 2 1 3

4 0 1 2

2 1 0 3

3 4 2 0

Kết quả mẫu:

7

In [ ]:

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 100

int n, c[MAX][MAX]; //# sô´ thành phô´ và ma trận chi phí
int cmin = INT_MAX; //# chi phí đi lại nhỏ nhâ´t giữa hai thành phô´ khác nhau
int best = INT_MAX; //# tổng chi phí nhỏ nhâ´t cần tìm, ban đầu đặt bằng
giá trị vô cùng lớn INT_MAX = 2^31-1
int curr; //# tổng chi phí tới thời điểm hiện tại
int mark[MAX]; //# đánh dâ´u những thành phô´ đã đi
int x[MAX]; //# lưu giữ các thành phô´ đã đi

//# Đọc dữ liệu vào
void input(){
    cin >> n;
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= n; ++j){
            cin >> c[i][j];
            if (c[i][j] > 0) cmin = min(cmin, c[i][j]);
        }
}

//# Thuật toán quay lui
void TRY(int k){
    for(int i = 2; i <= n; i++){
        /*****************
        # YOUR CODE HERE #
        *****************/
    }
}

int main() {
    input();
```

```
    x[1] = 1;
    TRY(2);
    cout << best;
    return 0;
}
```

1.3 Đệ quy có nhớ

Bài tập 4: LIS

Cho dãy a có n phần tử. Một dãy con của a là dãy thu được bằng cách xóa đi một số phần tử của a và giữ nguyên thứ tự các phần tử còn lại (có thể không xóa phần tử nào). Hãy tìm dãy con tăng dài nhất của a

Dữ liệu vào:

Dòng 1: Chứa số nguyên n ($1 \le n \le 1000$)

Dòng 2: Chứa n số nguyên $a_1\ a_2\ \dots\ a_n$ ($|a_i| \le 10^9$)

Kết quả:

Dòng đầu tiên chứa độ dài dãy con tăng dài nhất

Dòng thứ 2 chứa chỉ số các phần tử được chọn vào dãy con đó

Nếu có nhiều dãy con tăng dài nhất, in ra dãy bất kỳ trong số đó

Ví dụ:

Dữ liệu mẫu:

6

2 1 5 4 3 6

Kết quả mẫu:

3

2 5 6

Hướng dẫn:

Bài toán này được giải bằng phương pháp quy hoạch động.

Giả sử $lis(i)$ là độ dài dãy con tăng dài nhất kết thúc tại a_i. Khi đó ta có công thức truy hồi sau:

$$lis(i) = \max_{1 \le j \le i-1 : a_j < a_i} (lis(j) + 1)$$

In [ ]:

```
#include <bits/stdc++.h>
using namespace std;
int a[1000], n;
int mem[1000]; //# mảng ghi nhớ lời giải các bài toán con đã được giải

void init(){
    memset(mem, -1, sizeof(mem));
}

//# Quy hoạch động,
//# Hàm lis(i) trả về độ dài dãy con tăng dài nhất kết thúc bởi a[i]
int lis(int i) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}

//# Truy vet loi giai
void trace(int i){
    for(int j = 0; j < i; j++){
        if (a[j] < a[i] && mem[i] == 1 + mem[j]){
            trace(j);
```

```
            break;
        }
    }
    cout << a[i] << " ";
}

int main(){
    init();
    cin >> n;
    for(int i = 0; i < n; i++) cin >> a[i];
    int res = 1, pos = 0;
    for(int i = 1; i < n; i++){
        if (res < lis(i)){
            res = lis(i);
            pos = i;
        }
    }
    cout << res << endl;
    trace(pos);
    return 0;
}
```

Phần 2. Khử đệ quy

Hãy giải các bài toán sau đây bằng phương pháp khử đệ quy

Bài tập 5: Tính tổ hợp

Tính C_k^n

In [ ]:

```
#include <iostream>
using namespace std;

int binom(int n, int k) {
    if (k > n) return 0;
    if (k == 0) return 1;
    return binom(n-1, k) + binom(n-1, k-1);
}

int binom2(int n, int k){

    //# Khử đệ quy
    /*****************
    # YOUR CODE HERE #
    *****************/
}

int main() {
    int m;
    cin >> m;
    for (int n = 1; n <= m; ++n){
        for (int k = 0; k <= n; ++k)
            printf("%d ", binom(n, k));
        printf("\n");
    }
    for (int n = 1; n <= m; ++n){
        for (int k = 0; k <= n; ++k)
            printf("%d ", binom2(n, k));
        printf("\n");
    }
    return 0;
```

```
}
```

Bài tập 6: Tìm ước chung lớn nhất

Tính ước chung lớn nhất của hai số cho trước

In [ ]:

```
#include <iostream>
using namespace std;

int gcd(int a, int b){
    if (b == 0) return a;
    return gcd(b, a % b);
}

int gcd2(int a, int b){

    //# Khử đệ quy
    /*****************
    # YOUR CODE HERE #
    *****************/
}

int main() {
    int a, b;
    cin >> a >> b;
    cout << gcd(a, b) << endl << gcd2(a, b);
    return 0;
}
```

Bài tập 7: Liệt kê xâu nhị phân

Sử dụng phương pháp khử đệ quy bằng stack, hãy liệt kê các xâu nhị phân độ dài n không có k bit 1 nào liên tiếp

Dữ liệu vào:

Một dòng duy nhất chứa hai số nguyên n k ($1 \le k \le n \le 20$)

Kết quả:

Với mỗi xâu tìm được, in ra n ký tự trên một dòng, các ký tự cách nhau bởi dấu cách. Các xâu cần được liệt kê theo thứ tự từ điển

Ví dụ:

Dữ liệu mẫu:

4 2

Kết quả mẫu:

0 0 0 0
0 0 0 1
0 0 1 0
0 1 0 0
0 1 0 1
1 0 0 0
1 0 0 1
1 0 1 0

Lời giải đệ quy:

```
// Giả sử lời giải được lưu bởi xâu x1, x2, ..., xn
// i : biểu diễn lời giải bộ phận cấp i, trước đó x1, x2, ..., x[i-1] đã được gán giá trị
// j : giá trị ứng cử viên đang xét cho vị trí x[i]
// old_L: số ký tự 1 liên tiếp ở cuối dãy x1, x2, ..., x[i-1]
```

```
void TRY(int i, int j, int old_L){
    x[i] = j;
    if (i == n) {print_sol(); return;}
    int L = j ? ++old_L : 0;
    TRY(i + 1, 0, L);
    if (L + 1 < K) TRY(i + 1, 1, L);
}
```

In [ ]:

```
#include <bits/stdc++.h>
using namespace std;

struct state{
    int i, j, old_L;
    //# constructor
    state(int _i = 0, int _j = 0, int _L = 0):
        i(_i), j(_j), old_L(_L){}
};

int main() {
    int n, k;
    cin >> n >> k;
    int x[n+1];
    stack<state> s;
    //# number of consecutive suffix 1
    int L = 0;
    s.push(state(1, 0));
    while (!s.empty()){
        state &top = s.top();
        //# if a new binary sequence is found
        if (top.i > n){
            for (int i = 1; i <= n; ++i)
                cout << x[i] << " \n"[i == n];
            s.pop();
            continue;
        }

        //# Khử đệ quy
        /*****************
        # YOUR CODE HERE #
        *****************/
    }
    return 0;
}
```

Bài tập 8: Cân đĩa

Bạn đang muốn kiểm tra xem một vật cho trước có đúng nặng M như người ta nói hay không. Có một cân thăng bằng và n quả cân. Quả thứ i nặng mi. Hãy chỉ ra một cách cân thỏa mãn. Quy cách in ra đã được tích hợp trong mã nguồn dưới.

Dữ liệu mẫu:

6 10

7 1 2 3 4 5

Kết quả mẫu:

-1+2+3+4+5=10

In [ ]:

```cpp
#include <bits/stdc++.h>
using namespace std;

struct state{
    int i, j;
    state(int _i = 0, int _j = 0): i(_i), j(_j) {}
};

int main() {
    int n, M;
    cin >> n >> M;
    int m[n+1];
    for (int i = 1; i <= n; ++i) cin >> m[i];
    int x[n+1];
    stack<state> s;
    //# sum of selected weights
    int sum = 0;
    s.push(state(1, -1));
    while (!s.empty()){
        state &top = s.top();
        if (top.i > n){
            if (sum == M){
                for (int i = 1; i <= n; ++i){
                    if (x[i] == -1) cout << '-' << m[i];
                    if (x[i] == 1) cout << '+' << m[i];
                }
                cout << "=" << M;
                exit(0);
            }
            s.pop();
            continue;
        }

        //# Khử đệ quy
        /*****************
        # YOUR CODE HERE #
        *****************/
    }
    cout << -1;

    return 0;
}
```

Phần 3. Bài tập về nhà

Sinh viên tự làm các bài tập sau:

Bài tập 9: Lập lịch cho y tá

Một y tá cần lập lịch làm việc trong N ngày, mỗi ngày chỉ có thể là làm việc hay nghỉ ngơi. Một lịch làm việc là tốt nếu không có hai ngày nghỉ nào liên tiếp và mọi chuỗi ngày tối đại làm việc liên tiếp đều có số ngày thuộc đoạn $[K_1,K_2]$. Hãy liệt kê tất cả các cách lập lịch tốt, với mỗi lịch in ra trên một dòng một xâu nhị phân độ dài n với bit 0/1 tương ứng là nghỉ/làm việc. Các xâu phải được in ra theo thứ tự từ điển

Dữ liệu vào:

Ghi 3 số nguyên N,K_1,K_2 ($N \le 200, K_1 < K_2 \le 70$)

Kết quả:

Ghi danh sách các lịch tìm được theo thứ tự từ điển

Ví dụ:

Dữ liệu mẫu:
6 2 3
Kết quả mẫu:
011011
110110
110111
111011

Bài tập 10: Khoảng cách Hamming

Khoảng cách Hamming giữa hai xâu cùng độ dài là số vị trí mà ký tự tại vị trí đó là khác nhau trên hai xâu. Cho S là xâu gồm n ký tự 0. Hãy liệt kê tất cả các xâu nhị phân độ dài n, có khoảng cách Hamming với S bằng H. Các xâu phải được liệt kê theo thứ tự từ điển

Dữ liệu vào:

Dòng đầu chứa T là số testcase

T dòng tiếp theo, mỗi dòng mô tả một testcase, ghi N và H ($1 \le H \le N \le 16$)

Kết quả:

Với mỗi testcase, in ra danh sách các xâu thỏa mãn. In ra một dòng trống giữa hai testcase

Ví dụ:

Dữ liệu mẫu:
2
4 2
1 0
Kết quả mẫu:
0011
0101
0110
1001
1010
1100
0

Bài tập 11: Lịch trình chụp ảnh

Superior là một hòn đảo tuyệt đẹp với n địa điểm chụp ảnh và các đường một chiều nối các điểm chụp ảnh với nhau. Đoàn khách tham quan có r người với sở thích chụp ảnh khác nhau. Theo đó, mỗi người sẽ đưa ra danh sách các địa điểm mà họ muốn chụp. Bạn cần giúp mỗi người trong đoàn lập lịch di chuyển sao cho đi qua các điểm họ yêu cầu đúng một lần, không đi qua điểm nào khác, bắt đầu tại điểm đầu tiên và kết thúc tại điểm cuối cùng trong danh sách mà họ đưa ra, và có tổng khoảng cách đi lại là nhỏ nhất.

Dữ liệu vào:

Dòng đầu chứa n và r

Tiếp theo là ma trận $n \times n$ mô tả chi phí đi lại giữa các địa điểm. Chi phí bằng 0 có nghĩa là không thể đi lại giữa hai địa điểm đó.

r dòng tiếp theo chứa danh sách các địa điểm mà người r đưa ra. Lưu ý là hành mỗi hành trình cần phải bắt đầu và kết thúc bởi hai đỉnh đầu và cuối của danh sách, còn các địa điểm còn lại có thể thăm theo bất kỳ thứ tự nào

Kết quả:

Gồm r dòng ghi chi phí đi lại ít nhất của r người theo thứ tự đầu vào

Ví dụ:

Dữ liệu mẫu:

6 3
0 1 2 0 1 1
1 0 1 1 1 0
0 2 0 1 3 0
4 3 1 0 0 0
0 0 1 1 0 0
1 0 0 0 0 0
1 3 5
6 3 2 5
6 1 2 3 4 5

Kết quả mẫu:

5
0
7

Bài tập 12: Đếm đường đi

Cho đồ thị vô hướng G, hãy đếm số đường đi đi qua k cạnh và không đi qua đỉnh nào quá một lần.

Dữ liệu vào:

Dòng 1: Chứa hai số nguyên n và k ($1 \le n \le 30$, $1 \le k \le 10$) với n là số đỉnh của G. Các đỉnh sẽ được đánh số từ 1 đến n

Dòng 2: Chứa số nguyên m ($1 \le m \le 60$) là số cạnh của G

m dòng tiếp theo: Mỗi dòng chưa hai số nguyên là một cạnh của G

Kết quả:

Số lượng đường đi đơn độ dài k

Ví dụ:

Dữ liệu mẫu:

4 3
5
1 2
1 3
1 4
2 3
3 4

Kết quả mẫu:

6

In [ ]: