Sv. Huỳnh Vũ Nhật Linh | CN2304CLCB
Source Code: github.com/linhhuynhcoding/

Dynamic Programming (Quy Hoạch Động)

BÀI TOÁN KNAPSACK

> Có n đồ vật, vật thứ i có trọng lượng A_i và giá trị B_i. Hãy chọn ra một số các đồ vật, mỗi vật một cái để xếp vào 1 vali có trọng lượng tối đa W sao cho tổng giá trị của vali là lớn nhất.

```
int w;
cin >> n >> w;
vector<pair <int, int>> dovat(n);
for (auto &[u, v] : dovat) cin >> u;
for (auto &[u, v] : dovat) cin >> v;

int f[n + 1][w + 1];
for (int i = 0; i <= n; i++)
    f[i][0] = 0;
for (int i = 0; i <= w; i++)
    f[0][i] = 0;

for (int i = 1; i <= w; i++){
    for (int j = 1; j <= n; j++) {
        auto [x, v] = dovat[j - 1];
        f[j][i] = f[j - 1][i];
        if (x > i) continue;

        f[j][i] = max(f[j][i], f[j - 1][i - x] + v);
    }
}

cout << "Phuong an toi uu la : ";
cout << f[n][w];
```

```
5 40
46 40 42 38 10
12 19 19 15 8
Phuong an toi uu la : 19
```

DÃY CON CÓ TỔNG BẰNG S

> Cho dãy $A_1, A_2, \ldots, A_N$. Tìm một dãy con của dãy đó có tổng bằng S.

```
    int s;
    cin >> n >> s;
    vector <int> a(n);
    for (auto &i : a) cin >> i;
    bool f[n + 1][s + 1];
    int trace[s + 1];

    for (int i = 0; i <= s; i++)
        f[0][i] = false;
    for (int i = 0; i <= n; i++)
        f[i][0] = true;

    for (int i = 1; i <= s; i++){
        for (int j = 1; j <= n; j++){
            if (a[j - 1] > i) {
                f[j][i] = f[j - 1][i];
                continue;
            }
            if (f[j - 1][i - a[j - 1]]) {
                trace[i] = a[j - 1];
                f[j][i] = true;
            }
            else f[j][i] = f[j - 1][i] ;
        }
    }

    if (f[n][s]) cout << "co\n";
    else cout << "khong\n";
```

```
5 10
1 2 3 4 5
co
```

BÀI TẬP ỨNG DỤNG

BÀI 1

1. Một xâu gọi là xâu đối xứng (palindrome) nếu xâu đó đọc từ trái sang phải hay từ phải sang trái đều như nhau. Cho một xâu S, hãy tìm số kí tự ít nhất cần thêm vào S để S trở thành xâu đối xứng.

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

const int N = 5010;
int n, d[N][N];
string s;
```

```
int calc(int i, int j)
{
    if (d[i][j] == -1)
    {
        if (i >= j)
            d[i][j] = 0;
        else
        {
            if (s[i] == s[j])
                d[i][j] = calc(i + 1, j - 1);
            else
                d[i][j] = 1 + min(calc(i, j - 1), calc(i + 1, j));
        }
    }
    return d[i][j];
}

int main()
{
    cin >> s;
    n = s.length();
    s = "_" + s;
    for (int i = 0; i <= n; i++)
    for (int j = 0; j <= n; j++)
        d[i][j] = -1;
    cout << calc(1, n) << '\n';
}
```

BÀI 2

2. Ở đất nước Omega người ta chỉ tiêu tiền xu. Có N loại tiền xu, loại thứ i có mệnh giá là A_i đồng. Một người khách du lịch đến Omega du lịch với số tiền M đồng. Ông ta muốn đổi số tiền đó ra tiền xu Omega để tiện tiêu dùng. Ông ta cũng muốn số đồng tiền đổi được là ít nhất (cho túi tiền đỡ nặng khi đi đây đi đó). Bạn hãy giúp ông ta tìm cách đổi tiền.

```
    cin >> n >> m;
    vector<int> coins(n);

    for (auto &c : coins) cin >> c;
    int f[m + 1];

    for (int i = 1; i <= m; i++)
        f[i] = INF;

    // f[1] = 0;
    for (int i = 1; i <= m; i++) {
```

```
        for (auto c : coins) {
            if (c > i) {
                f[i] = 0;
                continue;
            }
            f[i] = min(f[i], f[i - c] + 1);
        }
    }

    cout << f[m];
```

BÀI 3

3. Có n cuộc họp, cuộc họp thứ i bắt đầu vào thời điểm A_i và kết thúc ở thời điểm B_i. Do chỉ có một phòng hội thảo nên 2 cuộc họp bất kì sẽ được cùng bố trí phục vụ nếu khoảng thời gian làm việc của chúng chỉ giao nhau tại đầu mút. Hãy bố trí phòng họp để phục vụ được nhiều cuộc họp nhất.

```
    cin >> n;
    vector<pair<int, int>> schedules(n);
    int fix_min = INF;
    int fix_max = -INF;

    for (auto &[s, f] : schedules){
        cin >> s >> f;
        fix_min = min(fix_min, f);
        fix_max = max(fix_max, f);
    }

    int dp[fix_max + 1];

    for (int i = 0; i <= fix_max; i++) dp[i] = 0;

    for (int i = 1; i <= fix_max; i++) {
        for (auto [s, f] : schedules) {
            // cout << s << " " << f << "\n";
            if (f != i){
                continue;
            }

            dp[i] = max(dp[i], dp[s] + 1);
        }
    }
    cout << dp[fix_max];
```

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