QUAY LUI, NHÁNH CẬN

[Bài 1. Tập hợp có tổng bằng S 2](#_heading=h.gjdgxs)

[Bài 2. In dãy số 3](#_heading=h.30j0zll)

[Bài 3. Dãy số 2 4](#_heading=h.1fob9te)

[Bài 4. Hoán vị xâu kí tự 5](#_heading=h.3znysh7)

[Bài 5. Di chuyển trong mê cung 5](#_heading=h.2et92p0)

[Bài 6. Dãy con có tổng bằng K 7](#_heading=h.tyjcwt)

[Bài 7. Tập con bằng nhau 8](#_heading=h.3dy6vkm)

[Bài 8. Di chuyển trong ma trận 8](#_heading=h.1t3h5sf)

[Bài 9. Bài toán N quân hậu 9](#_heading=h.4d34og8)

[Bài 10. Di chuyển trong mê cung 2 10](#_heading=h.2s8eyo1)

[Bài 11. Đổi chỗ chữ số 11](#_heading=h.3rdcrjn)

[Bài 12. Số nguyên tố 12](#_heading=h.26in1rg)

[Bài 13. Bài toán N quân hậu 2 13](#_heading=h.lnxbz9)

[Bài 14. Phân tích số 14](#_heading=h.35nkun2)

[Bài 15. Chia mảng 14](#_heading=h.1ksv4uv)

[Bài 16. Máy ATM 15](#_heading=h.44sinio)

[Bài 17. Người du lịch 16](#_heading=h.2jxsxqh)

[Bài 18. Từ điển 16](#_heading=h.z337ya)

[Bài 19. Biểu thức 17](#_heading=h.3j2qqm3)

[Bài 20. Đường đi dài nhất 18](#_heading=h.1y810tw)

[Bài 21. Số nhỏ nhất có N ước số 19](#_heading=h.4i7ojhp)

[Bài 22. Kí tự đặc biệt 20](#_heading=h.2xcytpi)

[Bài 23. Kí tự đặc biệt 21](#_heading=h.1ci93xb)

[Bài 24. Phân tích số 2 21](#_heading=h.3whwml4)

## Bài 1. Tập hợp có tổng bằng S

Xét tất cả các tập hợp các số nguyên dương có các phần tử khác nhau và không lớn hơn số n cho trước. Nhiệm vụ của bạn là hãy đếm xem có tất cả bao nhiêu tập hợp có số lượng phần tử bằng k và tổng của tất cả các phần tử trong tập hợp bằng s?

Các tập hợp là hoán vị của nhau chỉ được tính là một.

Ví dụ với n = 9, k = 3, s = 23, {6, 8, 9} là tập hợp duy nhất thỏa mãn.

Input:  Gồm nhiều bộ test (không quá 100 test).

Mỗi bộ test gồm 3 số nguyên n, k, s với 1 ≤ n ≤ 20, 1 ≤ k ≤ 10 và 1 ≤ s ≤ 155. Input kết thúc bởi 3 số 0.

Output:  Với mỗi test in ra số lượng các tập hợp thỏa mãn điều kiện đề bài.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 9 3 23  9 3 22  10 3 28  16 10 107  20 8 102  20 10 105  20 10 155  3 4 3  4 2 11  0 0 0 | 1  2  0  20  1542  5448  1  0  0 |

Source code : <https://ideone.com/ZlCWqi>

## Bài 2. In dãy số

Cho dãy số A[] gồm n số nguyên dương. Tam giác đặc biệt của dãy số A[] là tam giác được tạo ra bởi n hàng, trong đó hàng thứ 1 là dãy số A[], hàng i là tổng hai phần tử liên tiếp của hàng i-1 (2≤i≤n). Ví dụ A[] = {1, 2, 3, 4, 5}, khi đó tam giác được tạo nên như dưới đây:

[1, 2, 3, 4, 5 ]

[3, 5, 7, 9 ]

[8, 12, 16]

[20, 28]

[48]

            Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào N là số lượng phần tử của dãy số A[]; dòng tiếp theo đưa vào N số của mảng A[].
* T, N, A[i] thỏa mãn ràng buộc: 1≤T≤100; 1≤N, A[i] ≤10;

Output:

* Đưa ra tam giác tổng của mỗi test theo từng dòng. Mỗi dòng của tam giác tổng được bao bởi ký tự [, ].

|  |  |
| --- | --- |
| Input | Output |
| 1  5  1 2 3 4 5 | [1 2 3 4 5]  [3 5 7 9]  [8 12 16]  [20 28]  [48] |

Source code : <https://ideone.com/yLRqtR>

## Bài 3. Dãy số 2

Cho dãy số A[] gồm n số nguyên dương. Tam giác đặc biệt của dãy số A[] là tam giác được tạo ra bởi n hàng, trong đó hàng thứ n là dãy số A[], hàng i là tổng hai phần tử liên tiếp của hàng i+1 (1≤i≤n-1). Ví dụ A[] = {1, 2, 3, 4, 5}, khi đó tam giác được tạo nên như dưới đây:

[48]

[20, 28]

[8, 12, 16]

[3, 5, 7, 9 ]

[1, 2, 3, 4, 5 ]

            Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào N là số lượng phần tử của dãy số A[]; dòng tiếp theo đưa vào N số của mảng A[].
* T, N, A[i] thỏa mãn ràng buộc: 1≤T≤100; 1≤N, A[i] ≤10;

Output:

* Đưa ra kết quả mỗi test theo từng dòng. Mỗi dòng của tam giác tổng được bao bởi ký tự [, ].

|  |  |
| --- | --- |
| Input | Output |
| 1  5  1 2 3 4 5 | [48] [20 28] [8 12 16] [3 5 7 9 ] [1 2 3 4 5 ] |

Source code : <https://ideone.com/eFVqru>

## Bài 4. Hoán vị xâu kí tự

Cho xâu ký tự S bao gồm các ký tự in hoa khác nhau. Hãy đưa ra tất cả các hoán vị của xâu ký tự S. Ví dụ S=”ABC” ta có kết quả {ABC ACB BAC BCA CAB CBA}.

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test là một xâu ký tự S được viết trên 1 dòng.
* T, S thỏa mãn ràng buộc: 1≤T≤10; 1≤length(S) ≤10;

Output:

* Đưa ra kết quả mỗi test theo từng dòng.

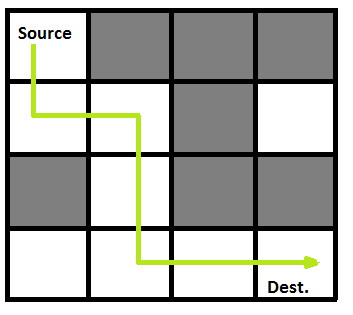
|  |  |
| --- | --- |
| Input | Output |
| 2 AB  ABC | AB BA  ABC ACB BAC BCA CAB CBA |

Source code : <https://ideone.com/8qvb3i>

## Bài 5. Di chuyển trong mê cung

Cho một mê cung bao gồm các khối được biểu diễn như một ma trận nhị phân A[N][N]. Một con chuột đi từ ô đầu tiên góc trái (A[0][0]) đến ô cuối cùng góc phải (A[N-1][N-1]) theo nguyên tắc:

* Down (D): Chuột được phép xuống dưới nếu ô dưới nó có giá trị 1.
* Right (R): Chuột được phép sang phải dưới nếu ô bên phải nó có giá trị 1.



Hãy đưa ra một hành trình của con chuột trên mê cung. Đưa ra -1 nếu chuột không thể đi đến đích.

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N là kích cỡ của mê cung; dòng tiếp theo đưa vào ma trận nhị phân A[N][N].
* T, N, A[i][j] thỏa mãn ràng buộc: 1≤T ≤10; 2≤N≤10; 0≤A[i][j] ≤1.

Output:

* Đưa ra tất cả đường đi của con chuột trong mê cung theo thứ tự từ điển. Đưa ra -1 nếu chuột không đi được đến đích.

|  |  |
| --- | --- |
| Input | Output |
| 2  4  1  0  0  0  1  1  0  1  0  1  0  0  1  1  1  1  5  1 0 0 0 0  1 1 1 1 1  1 1 0 0 1  0 1 1 1 1  0 0 0 1 1 | DRDDRR  DDRDRRDR DDRDRRRD DRDDRRDR DRDDRRRD DRRRRDDD |

Source code : <https://ideone.com/7R9tbR>

## Bài 6. Dãy con có tổng bằng K

Cho dãy số A[] = (a1, a2, .., an) và số tự nhiên K. Hãy đưa ra tất cả các dãy con của dãy số A[] sao cho tổng các phần tử của dãy con đó đúng bằng K. Các phần tử của dãy số A[] được giả thuyết là nguyên dương và không có các phần tử giống nhau. Ví dụ với dãy con A[] = {5, 10, 15, 20, 25}, K = 50 ta có 3 dãy con {5, 10, 15, 20}, {5, 20, 25}, {10, 15, 25}.

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N là số lượng phần tử của dãy số A[] và số K; dòng tiếp theo đưa vào N phần tử của dãy số A[].
* T, N, A[i] thỏa mãn ràng buộc: 1≤T ≤100; 1≤N≤10; 1≤ K, A[i] ≤100.

Output:

* Đưa ra tất cả các dãy con của dãy số A[] thỏa mãn yêu cầu bài toán theo thứ tự từ điển, trong đó mỗi dãy con được bao bởi các ký tự [, ]. Nếu không có dãy con nào thỏa mãn yêu cầu bài toán, hãy đưa ra -1.

|  |  |
| --- | --- |
| Input | Output |
| 2 5 50  5  10 15 20  25 8  53 15  22  14  26  32  9  16  8 | [5 10 15 20] [5 20 25] [10 15 25] [8 9 14 22] [8 14 15 16] [15 16 22] |

Source code : <https://ideone.com/scX40O>

## Bài 7. Tập con bằng nhau

Cho tập các số A[] = (a1, a2, .., an). Hãy kiểm tra xem ta có thể chia tập A[] thành hai tập con sao cho tổng các phần tử của hai tập con bằng nhau hay không. Đưa ra YES nếu có thể thực hiện được, ngược lại đưa ra NO.

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N là số lượng phần tử của dãy số A[]; dòng tiếp theo đưa vào N phần tử của dãy số A[].
* T, N, A[i] thỏa mãn ràng buộc: 1≤T ≤100; 1≤N≤100; 1≤ A[i] ≤100.

Output:

* Đưa ra kết quả mỗi test theo từng dòng.

|  |  |
| --- | --- |
| Input | Output |
| 2 4 1 5 11 5 3 1 3 5 | YES NO |

Source code : <https://ideone.com/38A6IJ>

## Bài 8. Di chuyển trong ma trận

Cho ma trận A[M][N]. Nhiệm vụ của bạn là đếm tất cả các đường đi từ phần tử A[0][0] đến phần tử A[M-1][N-1]. Bạn chỉ được phép dịch chuyển xuống dưới hoặc sang phải phần tử liền kề với vị trí hiện tại.

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số M, N tương ứng với số hàng và số cột của ma trận; dòng tiếp theo đưa vào các phần tử của ma trận A[][]; các số được viết cách nhau một vài khoảng trống.
* T, M, N, A[i][j] thỏa mãn ràng buộc: 1≤T ≤10; 1≤M, N, A[i][j]≤100.

Output:

* Đưa ra số cách di chuyển của mỗi test theo từng dòng.
* Giải thích test 1: Có 3 cách di chuyển là [1 4 5 6], [1 2 5 6] và [1 2 3 6].

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

Source code : <https://ideone.com/0ppriN>

## Bài 9. Bài toán N quân hậu

Cho một bàn cờ vua có kích thước n \* n, ta biết ràng quân hậu có thể di chuyển theo chiều ngang, dọc, chéo. Vấn đề đặt ra rằng, có n quân hậu, bạn cần đếm số cách đặt n quân hậu này lên bàn cờ sao cho với 2 quân hậu bất kì, chúng không “ăn” nhau.

Input: Dòng đầu ghi số bộ test T (T<5). Mỗi bộ test ghi một số nguyên dương n duy nhất (không quá 10)

Output:  Ghi kết quả mỗi bộ test trên một dòng. Số cách đặt quân hậu.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 1  4 | 2 |

Source code : <https://ideone.com/KQPLlt>

## Bài 10. Di chuyển trong mê cung 2

Cho một mê cung bao gồm các khối được biểu diễn như một ma trận nhị phân A[N][N]. Một con chuột đi từ ô đầu tiên góc trái (A[0][0]) đến ô cuối cùng góc phải (A[N-1][N-1]) theo nguyên tắc:

* Down (D): Chuột được phép xuống dưới nếu ô dưới nó có giá trị 1.
* Right (R): Chuột được phép sang phải dưới nếu ô bên phải nó có giá trị 1.
* Left (L): Chuột được phép sang trái dưới nếu ô bên trái nó có giá trị 1.
* Up (U): Chuột được phép lên trên nếu ô trên nó có giá trị 1.

Hãy đưa ra tất cả các hành trình của con chuột trên mê cung. Đưa ra -1 nếu chuột không thể đi đến đích.

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N là kích cỡ của mê cung; dòng tiếp theo đưa vào ma trận nhị phân A[N][N].
* T, N, A[i][j] thỏa mãn ràng buộc: 1≤T ≤10; 2≤N≤8; 0≤A[i][j] ≤1.

Output:

* Đưa ra các xâu ký tự được sắp xếp, trong đó mỗi xâu là một đường đi của con chuột trong mê cung. In ra đáp án theo thứ tự từ điển. Đưa ra -1 nếu chuột không đi được đến đích.

|  |  |
| --- | --- |
| Input | Output |
| 3 4 1 0 0 0  1 1 0 1  0 1 0 0  0 1 1 1 4 1 0 0 0  1 1 0 1  1 1 0 0  0 1 1 1  5  1 0 0 0 0  1 1 1 1 1  1 1 1 0 1  0 0 0 0 1  0 0 0 0 1 | DRDDRR  DDRDRR DRDDRR  DDRRURRDDD DDRURRRDDD DRDRURRDDD DRRRRDDD |

Source code : <https://ideone.com/2EtcyX>

## Bài 11. Đổi chỗ chữ số

Cho số tự nhiên K và xâu ký tự các chữ số S.  Nhiệm vụ của bạn là đưa ra số lớn nhất bằng cách thực hiện nhiều nhất K lần đổi chỗ các ký tự trong S. Ví dụ K =3 và S = “1234567” ta được “7654321”.

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là số K; dòng tiếp theo là xâu ký tự S.
* T, K, S thỏa mãn ràng buộc: 1≤T ≤100; 1≤K≤10; 1≤.lenght(S)≤7.

Output:

* Đưa ra kết quả mỗi test theo từng dòng.

|  |  |
| --- | --- |
| Input | Output |
| 3 4 1234567 3 3435335 2 1034 | 7654321 5543333 4301 |

Source code : <https://ideone.com/nP6VlK>

## Bài 12. Số nguyên tố

Cho ba số N, P, S. Trong đó, P là một số nguyên tố. Nhiệm vụ của bạn là đưa ra tất cả N số nguyên tố tính từ P + 1 có tổng bằng S. Ví dụ với S = 28, P=7, N =2 ta có kết quả 11 + 17 = 28. Với N = 3, P = 2, S = 23 ta có kết quả : {3, 7, 13}, {5, 7, 11}

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là bộ ba số S, P, N được viết trên một dòng.
* S, P, N thỏa mãn ràng buộc: 1≤T ≤100; 1 ≤ N ≤ 10; 2≤S, P≤200.

Output:

* Với mỗi test, dòng đầu tiên in ra số lượng đáp án tìm được. Mỗi dòng tiếp theo in ra kết quả tìm được theo thứ tự từ điển.

|  |  |
| --- | --- |
| Input | Output |
| 2 2  7  28 3  2  23 | 1  11 17  2  3 7 13  5 7 11 |

Source code : <https://ideone.com/xUGnAp>

## Bài 13. Bài toán N quân hậu 2

Cho một bàn cờ 8 x 8, mỗi ô có một giá trị A[i][j] nhất định (0 ≤ A[i][j] ≤ 100), tương ứng với điểm số đạt được nếu như bạn đặt một quân cờ vào đó.

Nhiệm vụ của bạn là đặt 8 quân hậu lên bàn cờ, sao cho không có 2 quân nào ăn nhau, và số điểm đạt được là lớn nhất.

Input: Dòng đầu tiên là số lượng bộ test T (T ≤ 20).

Mỗi test gồm 8 dòng, mỗi dòng 8 số nguyên mô tả bàn cờ.

Output:  Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 1  1 2 3 4 5 6 7 8  9 10 11 12 13 14 15 16  17 18 19 20 21 22 23 24  25 26 27 28 29 30 31 32  33 34 35 36 37 38 39 40  41 42 43 44 45 46 47 48  48 50 51 52 53 54 55 56  57 58 59 60 61 62 63 64 | 260 |

Source code : <https://ideone.com/GcdUat>

## Bài 14. Phân tích số

Cho mảng A[] gồm N số nguyên dương phân biệt và số X. Nhiệm vụ của bạn là tìm phép tổ hợp các số trong mảng A[] có tổng bằng X. Các số trong mảng A[] có thể được sử dụng nhiều lần. Mỗi tổ hợp các số của mảng A[] được in ra theo thứ tự không giảm các số. Ví dụ với A[] = {2, 4, 6, 8}, X = 8 ta có các tổ hợp các số như sau:

[2, 2, 2, 2], [2, 2, 4], [2, 6], [4, 4], [8].

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số N và X; dòng tiếp theo đưa vào N số của mmảng A[]; các số được viết cách nhau một vài khoảng trống.
* T, N, X, A[i] thỏa mãn ràng buộc: 1≤T ≤10; 1≤X, A[i]≤100. N ≤ 20.

Output:

* Đưa ra kết quả mỗi test theo từng dòng. Mỗi đường tổ hợp được bao bởi cặp ký tự [, ]. Đưa ra -1 nếu không có tổ hợp nào thỏa mãn yêu cầu bài toán.

|  |  |
| --- | --- |
| Input | Output |
| 1 4  8  2  4  6  8 | [2 2 2 2] [2 2 4] [2 6] [4 4] [8] |

Source code : <https://ideone.com/tcnuX9>

## Bài 15. Chia mảng

Cho mảng các số nguyên A[] gồm N phần tử. Hãy chia mảng số nguyên A[] thành K tập con khác rỗng sao cho tổng các phần tử của mỗi tập con đều bằng nhau. Mỗi phần tử thuộc tập con xuất hiện duy nhất một lần trong tất cả các tập con. Ví dụ với A[] = {2, 1, 4, 5, 6}, K =3 ta có kết quả {2, 4}, {1, 5}, {6}.

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số N và K; dòng tiếp theo đưa vào N số của mmảng A[]; các số được viết cách nhau một vài khoảng trống.
* T, N, K, A[i] thỏa mãn ràng buộc: 1≤T ≤100; 1≤N, K≤20, 0≤A[i]≤100.

Output:

* Đưa ra 1 nếu có thể chia tập con thành K tập thỏa mãn yêu cầu bài toán, ngược lại đưa ra 0.

|  |  |
| --- | --- |
| Input | Output |
| 2 5 3 2 1 4 5 6 5 3 2 1 5 5 6 | 1 0 |

Source code : <https://ideone.com/GZmNMZ>

## Bài 16. Máy ATM

Một máy ATM hiện có n (n ≤ 30) tờ tiền có giá trị t[1], t[2], …, t[n]. Hãy tìm cách trả ít tờ nhất với số tiền đúng bằng S (các tờ tiền có giá trị bất kỳ và có thể bằng nhau, mỗi tờ tiền chỉ được dùng một lần).

Input: Dòng đầu tiên ghi số bộ test T (T<10). Mỗi bộ test gồm 2 số nguyên n và S (S ≤ 109). Dòng thứ hai chứa n số nguyên t[1], t[2], …, t[n] (t[i] ≤ 109)

Output: Với mỗi bộ test ghi ra số tờ tiền ít nhất phải trả.

     Nếu không thể tìm được kết quả, in ra -1.

Ví dụ

|  |  |
| --- | --- |
| Input | Output |
| 1  3 5  1 4 5 | 1 |

Source code : <https://ideone.com/L8yLio>

## Bài 17. Người du lịch

Cho n thành phố đánh số từ 1 đến n và các tuyến đường giao thông hai chiều giữa chúng, mạng lưới giao thông này được cho bởi mảng C[1…n, 1…n] ở đây C[i][j] = C[j][i] là chi phí đi đoạn đường trực tiếp từ thành phố i đến thành phố j.

Một người du lịch xuất phát từ thành phố 1, muốn đi thăm tất cả các thành phố còn lại mỗi thành phố đúng 1 lần và cuối cùng quay lại thành phố 1. Hãy chỉ ra chi phí ít nhất mà người đó phải bỏ ra.

Dữ liệu vào: Dòng đầu tiên là số nguyên n – số thành phố (n ≤ 15); n dòng sau, mỗi dòng chứa n số nguyên thể hiện cho mảng 2 chiều C.

Kết quả: Chi phí mà người đó phải bỏ ra.

Ví dụ:

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 4  0 20 35 10  20 0 90 50  35 90 0 12  10 50 12 0 | 117 |

Source code : <https://ideone.com/qWLNxv>

## Bài 18. Từ điển

Cho tập từ ghi trong trừ điển dic[] và một bảng hai chiều A[M][N] các ký tự. Hãy tạo nên tất cả các từ có mặt trong từ điển dic[] bằng cách nối các ký tự kề nhau trong mảng A[][]. Chú ý, phép nối các ký tự kề nhau trong mảng A[][] được thực hiện theo 8 hướng nhưng không có phần tử A[i][j] nào được lặp lại. Ví dụ với từ điển dic[] ={ “GEEKS”, “FOR”, “QIUZ”, “GO”} và mảng A[][] dưới đây sẽ cho ta kết quả: “GEEKS”, “QUIZ”

|  |  |  |
| --- | --- | --- |
| G | I | Z |
| U | E | K |
| Q | S | E |

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào ba số K, M, N tương ứng với số từ của từ điển dic[], số hàng và số cột của ma trận ký tự A[M][N]; dòng tiếp theo đưa vào K từ của từ điển dic[]; dòng cuối cùng đưa vào các phần tử A[i][j].
* T, K, M, N thỏa mãn ràng buộc: 1≤T ≤10; 1≤K≤100; 1≤ M, N ≤3.

Output:

* Đưa ra theo thứ tự tăng dần các từ có mặt trong từ điển dic[] được tạo ra từ ma trận A[][]. Đưa ra -1 nếu không thể tạo ra từ nào thuộc dic[] từ A[][].

|  |  |
| --- | --- |
| Input | Output |
| 1 4  3  3 GEEKS FOR QUIZ GO G I Z  U E K  Q S E | GEEKS QUIZ |

Source code : <https://ideone.com/CMKFuh>

## Bài 19. Biểu thức

Cho 5 số nguyên dương A, B, C, D, E. Bạn có thể hoán vị các phần tử cho nhau, hãy đặt các dấu biểu thức +, -, \* sao cho biểu thức sau đúng:

[[[A o(1) B] o(2) C] o(3) D] o(4) E = 23

Trong đó: o(1) … o(4) là các phép toán +, -, \*.

Input:

Dòng đầu tiên là số lượng bộ test T (T ≤ 20).

Mỗi test gồm 5 số nguyên dương A, B, C, D, E có giá trị không vượt quá 100.

Output: Với mỗi test, in ra đáp án tìm được, mỗi xâu in ra trên một dòng.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 3  1 1 1 1 1  1 2 3 4 5  2 3 5 7 11 | NO  YES  YES |

Source code : <https://ideone.com/cdJxSg>

## Bài 20. Đường đi dài nhất

Cho đồ thị vô hướng có N đỉnh và M cạnh. Bạn hãy tìm đường đi dài nhất trên đồ thị, sao cho mỗi cạnh chỉ được đi qua nhiều nhất 1 lần.

Input: Dòng đầu tiên là số lượng bộ test T (T ≤ 10). Mỗi test bắt đầu bằng số nguyên N và M (1 ≤ N, M ≤ 20). Các đỉnh đánh dấu từ 0, 1, …, N-1. M dòng tiếp theo, mỗi dòng gồm 2 số u, v cho biết có cạnh nối giữa uàv.

Output: Với mỗi test, in ra đáp án tìm được, mỗi xâu in ra trên một dòng.

Ví dụ

|  |  |
| --- | --- |
| Input | Output |
| 2  3 2  0 1  1 2  15 16  0 2  1 2  2 3  3 4  3 5  4 6  5 7  6 8  7 8  7 9  8 10  9 11  10 12  11 12  10 13  12 14 | 2  12 |

Source code : <https://ideone.com/1oVV8r>

## Bài 21. Số nhỏ nhất có N ước số

Cho số nguyên dương N. Nhiệm vụ của bạn là tìm số K nhỏ nhất, sao cho K có đúng N ước. Input đảm bảo rằng đáp án không vượt quá 1018.

Input:

Dòng đầu tiên là số lượng bộ test T (T ≤  20).

Mỗi test gồm 1 số nguyên N ( 1 ≤  N ≤  1000).

Output:  Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 2  4  6 | 6  12 |

Source code : <https://ideone.com/GQOcFF>

## Bài 22. Kí tự đặc biệt

Cho một xâu s. Xâu F(s) được xác định bằng cách ghép xâu xâu s ban đầu với xâu s sau khi đã được quay vòng sang bên phải 1 kí tự (kí tự cuối cùng của s được chuyển lên đầu).

Thực hiện liên tiếp các bước cộng xâu như trên với xâu mới thu được, ta có được xâu X.

C:\Users\Xuan Loc\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\C85E5C64.tmp

Nhiệm vụ của bạn là hãy xác định kí tự thứ N trong xâu X là kí tự nào?

Input: Dòng đầu ghi số bộ test T (T<10). Mỗi bộ test gồm một xâu s có độ dài không vượt quá 30 kí tự và số nguyên N (1 ≤ N ≤ 1018).

Output: Với mỗi bộ test ghi ra trên một dòng kí tự tìm được.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 1  COW 8 | C |

Giải thích test: COW à COWWCO à COWWCOOCOWWC. Kí tự thứ 8 là ‘C’.

Source code : <https://ideone.com/1J5byD>

## Bài 23. Kí tự đặc biệt

Cho số nguyên dương N. Hãy đếm số bước ít nhất để đưa N về 1 bằng cách thực hiện ba thao tác dưới đây:

* Nếu N chia hết cho 2 bạn có thể giảm N = N/2.
* Nếu N chia hết cho 3 bạn có thể giảm N = N/3.
* Giảm N đi 1.

**Input:**

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số N được viết trên một dòng.
* T, N thỏa mãn ràng buộc: 1≤T≤100;  1≤N ≤100000.

**Output:**

* Đưa ra kết quả mỗi test theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 10 6 | 3  2 |

Source code : <https://ideone.com/DNUSrl>

## Bài 24. Phân tích số 2

Cho số nguyên dương N. Nhiệm vụ của bạn là hãy liệt kê tất cả các cách phân tích số tự nhiên N thành tổng các số tự nhiên nhỏ hơn hoặc bằng N. Phép hoán vị của một cách được xem là giống nhau. Ví dụ với N = 5 ta có kết quả là: (5), (4, 1), (3, 2), (3, 1, 1), (2, 2, 1), (2, 1, 1, 1), (1, 1, 1, 1, 1) .

**Input:**

* Dòng đầu tiên đưa vào số lượng test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số tự nhiên N được viết trên một dòng.
* T, n thỏa mãn ràng buộc: 1≤T, N≤10.

**Output:**

* Dòng đầu tiên là số lượng cách phân tích thỏa mãn. Dòng tiếp theo liệt kê đáp án theo mẫu ví dụ đã cho.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  4  5 | 5  (4) (3 1) (2 2) (2 1 1) (1 1 1 1)  7  (5) (4 1) (3 2) (3 1 1) (2 2 1) (2 1 1 1) (1 1 1 1 1) |

Source code : <https://ideone.com/te8s8T>

**1. Ghi nhớ kết quả đã tính (Memoization)**

**Bài toán: Đường đi tối thiểu trong tam giác số**

Cho một tam giác số:

2

3 4

6 5 7

4 1 8 3

Hãy tìm đường đi từ đỉnh đến đáy sao cho tổng các số trên đường đi là nhỏ nhất. Mỗi bước, bạn chỉ có thể đi xuống dưới (cùng cột hoặc cột bên phải).

**Cách giải:**

Sử dụng **quay lui + ghi nhớ** để tránh tính lại kết quả cho các trạng thái (i, j) (từ hàng i, cột j đến đáy).

def min\_path(i, j):

# Nếu đến đáy tam giác, trả về giá trị tại vị trí hiện tại

if i == len(triangle) - 1:

return triangle[i][j]

# Nếu đã tính toán, trả về kết quả đã lưu

if memo[i][j] != -1:

return memo[i][j]

# Tính tổng nhỏ nhất bằng cách đi xuống dưới hoặc xuống chéo

memo[i][j] = triangle[i][j] + min(min\_path(i + 1, j), min\_path(i + 1, j + 1))

return memo[i][j]

# Dữ liệu tam giác

triangle = [

[2],

[3, 4],

[6, 5, 7],

[4, 1, 8, 3]

]

memo = [[-1] \* len(row) for row in triangle]

print(min\_path(0, 0)) # Output: 11

**2. Dừng nhánh sớm (Pruning)**

**Bài toán: Sudoku Solver**

Cho một bảng Sudoku chưa hoàn chỉnh (9x9), hãy hoàn thành bảng bằng cách điền các số từ 1-9 sao cho:

* Mỗi hàng, mỗi cột, và mỗi ô vuông 3x3 không có số nào bị lặp lại.

**Cách giải:**

Sử dụng **quay lui với kiểm tra ràng buộc sớm**:

* Dừng nhánh nếu số vừa điền vi phạm quy tắc Sudoku.

def is\_valid(board, row, col, num):

# Kiểm tra hàng, cột và ô 3x3

for i in range(9):

if board[row][i] == num or board[i][col] == num:

return False

box\_row, box\_col = row // 3 \* 3, col // 3 \* 3

for i in range(box\_row, box\_row + 3):

for j in range(box\_col, box\_col + 3):

if board[i][j] == num:

return False

return True

def solve\_sudoku(board):

for i in range(9):

for j in range(9):

if board[i][j] == 0: # Nếu ô trống

for num in range(1, 10):

if is\_valid(board, i, j, num): # Kiểm tra tính hợp lệ

board[i][j] = num

if solve\_sudoku(board): # Thử tiếp

return True

board[i][j] = 0 # Quay lui

return False

return True

# Input bảng Sudoku

board = [

[5, 3, 0, 0, 7, 0, 0, 0, 0],

[6, 0, 0, 1, 9, 5, 0, 0, 0],

[0, 9, 8, 0, 0, 0, 0, 6, 0],

[8, 0, 0, 0, 6, 0, 0, 0, 3],

[4, 0, 0, 8, 0, 3, 0, 0, 1],

[7, 0, 0, 0, 2, 0, 0, 0, 6],

[0, 6, 0, 0, 0, 0, 2, 8, 0],

[0, 0, 0, 4, 1, 9, 0, 0, 5],

[0, 0, 0, 0, 8, 0, 0, 7, 9]

]

solve\_sudoku(board)

print(board) # Output: Bảng Sudoku hoàn chỉnh

**3. Giới hạn ràng buộc (Constraint Propagation)**

**Bài toán: N quân hậu**

Đặt N quân hậu trên bàn cờ N x N sao cho không có 2 quân nào "ăn" nhau.

**Cách giải:**

Áp dụng ràng buộc:

* Mỗi hàng chỉ có thể chứa 1 quân hậu.
* Đặt một quân hậu tại (i, j) sẽ loại bỏ tất cả các ô cùng cột và đường chéo của (i, j).

**Mã Python:**

def is\_safe(board, row, col):

for i in range(row):

if board[i] == col or abs(board[i] - col) == abs(i - row):

return False

return True

def n\_queens(board, row):

if row == len(board):

solutions.append(board[:])

return

for col in range(len(board)):

if is\_safe(board, row, col):

board[row] = col

n\_queens(board, row + 1)

board[row] = -1

N = 8

board = [-1] \* N

solutions = []

n\_queens(board, 0)

print(len(solutions)) # Output: Số lượng cách đặt N quân hậu

Dưới đây là **5 bài toán mức độ trung bình - khó (Medium Hard)**, mỗi bài toán ứng dụng một trong các kỹ thuật tối ưu mà em đã liệt kê. Em cũng cung cấp cách giải cụ thể cho từng bài nhé:

**1. Ghi nhớ kết quả đã tính (Memoization)**

**Bài toán: Đường đi tối thiểu trong tam giác số**

Cho một tam giác số:

markdown

Sao chép mã

2

3 4

6 5 7

4 1 8 3

Hãy tìm đường đi từ đỉnh đến đáy sao cho tổng các số trên đường đi là nhỏ nhất. Mỗi bước, bạn chỉ có thể đi xuống dưới (cùng cột hoặc cột bên phải).

**Cách giải:**

Sử dụng **quay lui + ghi nhớ** để tránh tính lại kết quả cho các trạng thái (i, j) (từ hàng i, cột j đến đáy).

**Mã Python:**

python

Sao chép mã

def min\_path(i, j):

# Nếu đến đáy tam giác, trả về giá trị tại vị trí hiện tại

if i == len(triangle) - 1:

return triangle[i][j]

# Nếu đã tính toán, trả về kết quả đã lưu

if memo[i][j] != -1:

return memo[i][j]

# Tính tổng nhỏ nhất bằng cách đi xuống dưới hoặc xuống chéo

memo[i][j] = triangle[i][j] + min(min\_path(i + 1, j), min\_path(i + 1, j + 1))

return memo[i][j]

# Dữ liệu tam giác

triangle = [

[2],

[3, 4],

[6, 5, 7],

[4, 1, 8, 3]

]

memo = [[-1] \* len(row) for row in triangle]

print(min\_path(0, 0)) # Output: 11

**2. Dừng nhánh sớm (Pruning)**

**Bài toán: Sudoku Solver**

Cho một bảng Sudoku chưa hoàn chỉnh (9x9), hãy hoàn thành bảng bằng cách điền các số từ 1-9 sao cho:

* Mỗi hàng, mỗi cột, và mỗi ô vuông 3x3 không có số nào bị lặp lại.

**Cách giải:**

Sử dụng **quay lui với kiểm tra ràng buộc sớm**:

* Dừng nhánh nếu số vừa điền vi phạm quy tắc Sudoku.

**Mã Python:**

python

Sao chép mã

def is\_valid(board, row, col, num):

# Kiểm tra hàng, cột và ô 3x3

for i in range(9):

if board[row][i] == num or board[i][col] == num:

return False

box\_row, box\_col = row // 3 \* 3, col // 3 \* 3

for i in range(box\_row, box\_row + 3):

for j in range(box\_col, box\_col + 3):

if board[i][j] == num:

return False

return True

def solve\_sudoku(board):

for i in range(9):

for j in range(9):

if board[i][j] == 0: # Nếu ô trống

for num in range(1, 10):

if is\_valid(board, i, j, num): # Kiểm tra tính hợp lệ

board[i][j] = num

if solve\_sudoku(board): # Thử tiếp

return True

board[i][j] = 0 # Quay lui

return False

return True

# Input bảng Sudoku

board = [

[5, 3, 0, 0, 7, 0, 0, 0, 0],

[6, 0, 0, 1, 9, 5, 0, 0, 0],

[0, 9, 8, 0, 0, 0, 0, 6, 0],

[8, 0, 0, 0, 6, 0, 0, 0, 3],

[4, 0, 0, 8, 0, 3, 0, 0, 1],

[7, 0, 0, 0, 2, 0, 0, 0, 6],

[0, 6, 0, 0, 0, 0, 2, 8, 0],

[0, 0, 0, 4, 1, 9, 0, 0, 5],

[0, 0, 0, 0, 8, 0, 0, 7, 9]

]

solve\_sudoku(board)

print(board) # Output: Bảng Sudoku hoàn chỉnh

**3. Giới hạn ràng buộc (Constraint Propagation)**

**Bài toán: N quân hậu**

Đặt N quân hậu trên bàn cờ N x N sao cho không có 2 quân nào "ăn" nhau.

**Cách giải:**

Áp dụng ràng buộc:

* Mỗi hàng chỉ có thể chứa 1 quân hậu.
* Đặt một quân hậu tại (i, j) sẽ loại bỏ tất cả các ô cùng cột và đường chéo của (i, j).

**Mã Python:**

python

Sao chép mã

def is\_safe(board, row, col):

for i in range(row):

if board[i] == col or abs(board[i] - col) == abs(i - row):

return False

return True

def n\_queens(board, row):

if row == len(board):

solutions.append(board[:])

return

for col in range(len(board)):

if is\_safe(board, row, col):

board[row] = col

n\_queens(board, row + 1)

board[row] = -1

N = 8

board = [-1] \* N

solutions = []

n\_queens(board, 0)

print(len(solutions)) # Output: Số lượng cách đặt N quân hậu

**4. Sắp xếp thứ tự lựa chọn (Order of Choices)**

**Bài toán: Tìm đường đi ngắn nhất trong mê cung**

Cho mê cung M x N với 1 là ô đi được và 0 là ô bị chặn. Hãy tìm đường đi từ (0, 0) đến (M-1, N-1).

**Cách giải:**

* Thử các hướng đi gần đích trước (ưu tiên giảm khoảng cách Manhattan).

**Mã Python:**

def heuristic(x, y):

return abs(M-1 - x) + abs(N-1 - y) # Khoảng cách Manhattan

def find\_path(x, y, path):

if x == M-1 and y == N-1:

results.append(path)

return

for dx, dy, move in sorted([(0, 1, 'R'), (1, 0, 'D')], key=lambda d: heuristic(x+d[0], y+d[1])):

nx, ny = x + dx, y + dy

if 0 <= nx < M and 0 <= ny < N and maze[nx][ny] == 1:

maze[x][y] = 0

find\_path(nx, ny, path + move)

maze[x][y] = 1

M, N = 3, 3

maze = [

[1, 1, 0],

[1, 1, 1],

[0, 1, 1]

]

results = []

find\_path(0, 0, "")

print(results) # Output: Các đường đi ngắn nhất

**5. Kết hợp với thuật toán khác**

**Bài toán: Tổ hợp con có tổng bằng S**

Cho mảng A[] và một số S. Hãy tìm tất cả các tổ hợp con sao cho tổng của chúng bằng S.

**Cách giải:**

Kết hợp quay lui và quy hoạch động để tối ưu:

* Dùng quay lui để sinh tổ hợp con.
* Kiểm tra tổng của tổ hợp con bằng cách lưu trước các trạng thái.

def find\_combinations(index, current, total):

if total == S:

results.append(current[:])

return

if total > S or index == len(A):

return

find\_combinations(index + 1, current + [A[index]], total + A[index])

find\_combinations(index + 1, current, total)

A = [2, 3, 6, 7]

S = 7

results = []

find\_combinations(0, [], 0)

print(results) # Output: [[7], [2, 3, 2]]

**Bài 1: Tập con không liền kề có tổng lớn nhất**

**Mô tả:**

Cho một mảng số nguyên A[] gồm N phần tử. Hãy tìm tập con có tổng lớn nhất sao cho không có hai phần tử nào trong tập con nằm kề nhau trong mảng.

**Ví dụ:**

* **Input**:

css

Sao chép mã

N = 5

A = [3, 2, 5, 10, 7]

* **Output**:

Sao chép mã

15

Tập con tối ưu: [3, 10].

**Gợi ý giải:**

1. Dùng quay lui để kiểm tra từng tập con:
   * Tại mỗi phần tử, anh có hai lựa chọn:
     + Chọn phần tử đó và bỏ qua phần tử kế tiếp.
     + Bỏ qua phần tử hiện tại.
2. Giữ lại tổng lớn nhất trong quá trình quay lui.

**Bài 2: Tổ hợp con có tổng bằng S với số phần tử cố định**

**Mô tả:**

Cho một mảng số nguyên dương A[] gồm N phần tử và một số nguyên S. Hãy tìm tất cả các tổ hợp con gồm đúng K phần tử sao cho tổng của chúng bằng S.

**Ví dụ:**

* **Input**:

css

Sao chép mã

N = 5, K = 3, S = 10

A = [1, 2, 3, 4, 5]

* **Output**:

lua

Sao chép mã

[[1, 2, 7], [2, 3, 5]]

**Gợi ý giải:**

1. Sử dụng quay lui để sinh tất cả các tổ hợp con gồm K phần tử.
2. Kiểm tra điều kiện tổng bằng S trước khi lưu kết quả.

**Bài 3: Phân chia mảng thành hai tập con có tổng bằng nhau**

**Mô tả:**

Cho một mảng số nguyên dương A[] gồm N phần tử. Hãy kiểm tra xem có thể phân chia mảng thành hai tập con sao cho tổng các phần tử của hai tập con bằng nhau hay không.

**Ví dụ:**

* **Input**:

css

Sao chép mã

N = 4

A = [1, 5, 11, 5]

* **Output**:

graphql

Sao chép mã

True

Tập con [1, 5, 5] và [11] có tổng bằng nhau.

**Gợi ý giải:**

1. Tính tổng sum của mảng. Nếu sum lẻ, trả về False.
2. Sử dụng quay lui để kiểm tra xem có tập con nào có tổng bằng sum // 2.

**Bài 4: Đặt quân mã trên bàn cờ**

**Mô tả:**

Cho một bàn cờ kích thước N x N. Đặt quân mã ở ô (0, 0) và di chuyển nó sao cho nó đi qua tất cả các ô trên bàn cờ đúng một lần.

**Ví dụ:**

* **Input**:

makefile

Sao chép mã

N = 5

* **Output**:

less

Sao chép mã

Một hành trình hoàn chỉnh, ví dụ:

[ [0, 1, 2, 3, 4],

[19, 18, 17, 16, 5],

...

]

**Gợi ý giải:**

1. Sử dụng quay lui để thử tất cả các nước đi hợp lệ của quân mã.
2. Kiểm tra:
   * Không đi ra ngoài bàn cờ.
   * Không đi vào ô đã được thăm.
3. Dừng khi đi qua đủ N \* N ô.

**Bài 5: Mê cung với nhiều lối thoát**

**Mô tả:**

Cho một mê cung kích thước M x N, trong đó:

* 1 đại diện cho đường đi.
* 0 đại diện cho tường. Hãy tìm tất cả các đường đi từ ô bắt đầu (0, 0) đến tất cả các ô ở hàng cuối cùng (M-1, j) (với mọi j hợp lệ).

**Ví dụ:**

* **Input**:

csharp

Sao chép mã

M = 3, N = 3

maze = [

[1, 1, 0],

[1, 1, 1],

[0, 1, 1]

]

* **Output**:

css

Sao chép mã

['RRDD', 'RDDR', 'DRDR', 'DRRD']

**Gợi ý giải:**

1. Sử dụng quay lui để thử tất cả các hướng đi từ (0, 0):
   * Đi xuống.
   * Đi sang phải.
2. Lưu lại các đường đi khi đạt đến hàng cuối cùng (M-1).

**Tóm tắt:**

1. **Tập con không liền kề có tổng lớn nhất**: Dùng quay lui để kiểm tra chọn/bỏ từng phần tử.
2. **Tổ hợp con có tổng bằng S với số phần tử cố định**: Sinh tổ hợp và kiểm tra điều kiện tổng.
3. **Phân chia mảng thành hai tập con có tổng bằng nhau**: Kiểm tra tập con với tổng bằng sum // 2.
4. **Đặt quân mã trên bàn cờ**: Thử tất cả các nước đi hợp lệ của quân mã.
5. **Mê cung với nhiều lối thoát**: Liệt kê tất cả các đường đi từ (0, 0) đến hàng cuối.

Nếu anh muốn lời giải chi tiết hoặc mã Python cụ thể cho bất kỳ bài nào, em sẵn sàng hỗ trợ thêm nhé! 😊

**Bạn đã nói:**

hãy cho anh code của 5 bài trên có áp dụng kỹ thuật tối ưu nha

**ChatGPT đã nói:**

Dưới đây là mã Python chi tiết cho **5 bài toán**, mỗi bài đều áp dụng các kỹ thuật tối ưu như **memoization**, **pruning**, **constraint propagation**, và các kỹ thuật liên quan.

**Bài 1: Tập con không liền kề có tổng lớn nhất**

**Mã Python (Memoization):**

python

Sao chép mã

def max\_non\_adjacent\_sum(i):

if i >= len(A):

return 0

if memo[i] != -1:

return memo[i]

# Chọn hoặc bỏ phần tử hiện tại

pick = A[i] + max\_non\_adjacent\_sum(i + 2)

skip = max\_non\_adjacent\_sum(i + 1)

memo[i] = max(pick, skip)

return memo[i]

# Input

A = [3, 2, 5, 10, 7]

memo = [-1] \* len(A)

print(max\_non\_adjacent\_sum(0)) # Output: 15

**Bài 2: Tổ hợp con có tổng bằng S với số phần tử cố định**

**Mã Python (Pruning):**

python

Sao chép mã

def find\_combinations(index, current, total, count):

if count == K and total == S:

results.append(current[:])

return

if count > K or total > S or index == len(A):

return

# Chọn phần tử hiện tại

find\_combinations(index + 1, current + [A[index]], total + A[index], count + 1)

# Bỏ qua phần tử hiện tại

find\_combinations(index + 1, current, total, count)

# Input

A = [1, 2, 3, 4, 5]

K, S = 3, 10

results = []

find\_combinations(0, [], 0, 0)

print(results) # Output: [[1, 4, 5], [2, 3, 5]]

**Bài 3: Phân chia mảng thành hai tập con có tổng bằng nhau**

**Mã Python (Memoization):**

python

Sao chép mã

def can\_partition(index, target):

if target == 0:

return True

if index >= len(A) or target < 0:

return False

if (index, target) in memo:

return memo[(index, target)]

# Chọn hoặc bỏ phần tử hiện tại

memo[(index, target)] = can\_partition(index + 1, target - A[index]) or can\_partition(index + 1, target)

return memo[(index, target)]

# Input

A = [1, 5, 11, 5]

total = sum(A)

if total % 2 != 0:

print(False) # Output: False

else:

memo = {}

print(can\_partition(0, total // 2)) # Output: True

**Bài 4: Đặt quân mã trên bàn cờ**

**Mã Python (Constraint Propagation):**

python

Sao chép mã

def is\_valid(x, y):

return 0 <= x < N and 0 <= y < N and board[x][y] == -1

def knight\_tour(x, y, move):

if move == N \* N:

return True

for dx, dy in moves:

nx, ny = x + dx, y + dy

if is\_valid(nx, ny):

board[nx][ny] = move

if knight\_tour(nx, ny, move + 1):

return True

board[nx][ny] = -1 # Quay lui

return False

# Input

N = 5

board = [[-1] \* N for \_ in range(N)]

moves = [(-2, -1), (-1, -2), (1, -2), (2, -1), (2, 1), (1, 2), (-1, 2), (-2, 1)]

board[0][0] = 0

if knight\_tour(0, 0, 1):

for row in board:

print(row)

else:

print("No solution")

**Bài 5: Mê cung với nhiều lối thoát**

**Mã Python (Pruning + Order of Choices):**

python

Sao chép mã

def find\_paths(x, y, path):

if x == M - 1:

results.append(path)

return

for dx, dy, move in [(0, 1, 'R'), (1, 0, 'D')]:

nx, ny = x + dx, y + dy

if 0 <= nx < M and 0 <= ny < N and maze[nx][ny] == 1:

maze[x][y] = 0 # Đánh dấu đã thăm

find\_paths(nx, ny, path + move)

maze[x][y] = 1 # Quay lui

# Input

maze = [

[1, 1, 0],

[1, 1, 1],

[0, 1, 1]

]

M, N = len(maze), len(maze[0])

results = []

find\_paths(0, 0, "")

print(results) # Output: ['RRDD', 'RDDR', 'DRDR', 'DRRD']

**Tóm tắt kỹ thuật áp dụng:**

1. **Bài 1 (Ghi nhớ kết quả đã tính)**:
   * Memoization giúp tránh tính lại các tổng cho những chỉ số đã xét.
2. **Bài 2 (Dừng nhánh sớm)**:
   * Khi số phần tử hoặc tổng vượt quá yêu cầu, dừng nhánh ngay lập tức.
3. **Bài 3 (Ghi nhớ kết quả)**:
   * Lưu trạng thái (index, target) để tránh tính toán lại cùng một tổng.
4. **Bài 4 (Ràng buộc quân mã)**:
   * Không di chuyển đến ô đã thăm hoặc ngoài bàn cờ.
5. **Bài 5 (Dừng nhánh sớm và ưu tiên hướng đi)**:
   * Dừng nếu không thể tiếp tục và ưu tiên thử hướng đi hợp lý.

Nếu anh cần giải thích kỹ hơn hoặc điều chỉnh bài toán, em sẵn sàng hỗ trợ nhé! 😊

Top of Form

Bottom of Form