SPOJ.COM – Thuật toán bài LKS – Large Knapsack

l lampv606 | } 29/10/2016 | v 3 bình luận

m Quy hoạch động - Dynamic programming, Spoj

phạm văn lâm, quy hoạch động dynamic programming, spoj.com, thuật toán, thuật toán spoj.com

Đề bài:

Bài toán knapsack hay rucksack là một bài toán trong combinatorial optimization:

Cho một tập gồm những phần tử. mỗi phần tử bao gồm trọng lượng và giá trị. Xác định số lượng mỗi phần tử được chọn sao cho tổng trọng lượng nhỏ hơn hoặc bằng với giới hạn cho trước và tổng giá trị là lớn nhất có thể.

Đầu vào:

Dòng đầu tiên bao gồm 2 số nguyên K và N, trong đó K là kích thước giới hạn của cái túi (knapsack) và N là số lượng phần tử. N dòng tiếp theo trong đó dòng thứ i là phần tử thứ i với giá trị v_i và trọng lượng w_i.

Đầu ra:

In ra số nguyên duy nhất, là giá trị lớn nhất có thể của chiếc túi. (Tất cả các phép toán đảm bảo trong phạm vi của số nguyên có dấu 32 bit)

Ví dụ:

Đầu vào:

10 3
7 3
8 8
4 6

Đầu ra:

11

Giới hạn:

K <= 2000000

N <= 500

V_i <= 10^7

W_i <= 10^7

Các bạn có thể tham khảo link gốc đề bài và submit code tại đây: http://www.spoj.com/problems/LKS/

Phân tích:

+ Đây là bài toán Knapsack 0/1 cơ bản. Nghĩa là mỗi đồ vật sẽ có thể được chọn hay không được chọn. Và nếu được chọn thì không giới hạn số lần chọn. Thuật toán được sử dụng ở đây là thuật toán quy hoạch động Dynamic Programming

+ Bình thường thì tôi sẽ sử dụng một mảng hai chiều kích thước NxK, là MaxCost[N][K]. Trong đó, MaxCost[i][j] là giá trị lớn nhất thu được với việc chọn đồ vật từ 1 đến i và trọng lượng không vượt quá j.

+ Nhưng do số lượng đầu vào khá lớn nên tôi không thể sử dụng một mảng 2 chiều NxK được. Tuy nhiên, nếu các bạn để ý rằng giá trị tại hàng thứ i chỉ phụ thuộc vào hàng trước đó i – 1 (thuật toán các bạn sẽ thấy rõ hơn ở dưới). Nên tôi chỉ cần một mảng 2 chiều 2xK, trong đó một hàng dùng để lưu giá trị hàng thứ i – 1 và một hàng lưu giá trị hàng i. Sau khi tính được hàng i, tôi lại hoán đổi vai trò của hai hàng cho nhau. Cứ như vậy, tôi sẽ tính được giá trị của hàng thứ N.

Lời giải:

(Các bạn nên tự mình nghĩ ra thuật toán của bài toán trước khi tham khảo code của tôi nhé. Hãy phát huy tối đa khả năng sáng tạo của bản thân. Hơn nữa code tôi viết ra cũng chưa thật sự tối ưu. Nên rất mong nhận được sự chia sẻ của các bạn.)

Code C/ C++:

```
1. #include <iostream>
2. using namespace std;
3.
```

```
const int MAX_N = 505;
const int MAX_K = 2000005;

int K, N;
int Value[MAX_N], Weight[MAX_N];
int MaxCost[2][MAX_K];
// MAX[1][j] là giá trị lớn nhất thu được
// với việc chọn từ đồ vật từ 1 đến i và
// khối lượng không vượt quá j

// MAX[0][j] là giá trị cũ trước đó.
// Và ngược lại

int Max(int a, int b)
{
    if(a > b) return a;
    return b;
}

int main()
{
    int T, test_case;
    ios::sync_with_stdio(false);
    //freopen("input.txt", "r", stdin);

    cin >> K >> N;
    for(int i = 1; i <= N; i++)
        cin >> Value[i] >> Weight[i];

    // Trường hợp cơ sở
    for(int j = 0; j <= K; j++)
        MaxCost[0][j] = 0;
```

```
    for(int i = 0; i < 2; i++)
        MaxCost[i][0] = 0;

    int before = 0, current = 1;
    for(int i = 1; i <= N; i++)
    {
        for(int j = 1; j <= K; j++)
        {
            // Không chọn vật i
            MaxCost[current][j] = MaxCost[before][j];

            // Chọn vật i
            if(Weight[i] <= j) MaxCost[current][j] =
            Max(MaxCost[current][j], MaxCost[before][j - Weight[i]] + Value[i]);
        }

        // Đổi current và before cho nhau.
        current = 1 - current;
        before  = 1 - before;
    }
    // N chẵn thì kết quả là MaxCost[0][K]
    // N lẻ thì kết quả là MaxCost[1][K]
    cout << MaxCost[N % 2][K] << endl;

    return 0;//Your program should return 0 on normal termination.
}
```

Code by Phạm Văn Lâm.

Bài viết liên quan

- SPOJ.COM – Thuật toán bài BYTESM2 – Philosophers Stone
- SPOJ.COM – Thuật toán bài ABA12C – Buying Apples
- SPOJ.COM – Thuật toán bài COINS – Bytelandian gold coins
- SPOJ.COM – Thuật toán bài AFS – Amazing Factor Sequence
- SPOJ.COM – Thuật toán bài ANARC05B – The Double HeLiX
- SPOJ.COM – Thuật toán bài ADFRUITS – Advanced Fruits
- SPOJ.COM – Thuật toán bài ACODE – Alphacode
- SPOJ.COM – Thuật toán bài DIEHARD – Die Hard Game

---

399 views

← Bài viết trước Bài tiếp theo →

3 bình luận

Phạm Văn Lâm 29/10/2016 at 9:27 Chiều

Trên đây là đề bài và lời giải bài LKS – Large Knapsack trên trang spoj.com. Rất mong nhận được sự ủng hộ và phản hồi từ các bạn.

Trân trọng,

Phạm Văn Lâm.

Cường 18/10/2017 at 12:45 Sáng

Cho mình hỏi tại sao cái trường cơ sở lại cho nó bằng 0

lampv606 (Post author) 18/10/2017 at 8:22 Chiều

Như mình đã giải thích:

// MAX[1][j] là giá trị lớn nhất thu được

// với việc chọn từ đồ vật từ 1 đến i và

// khối lượng không vượt quá j

Do đó nếu i, hoặc j == 0 thì giá trị cơ sở đều = 0. Ví dụ, j = 0, nghĩa là chọn làm sao cho khối lượng không vượt quá 0. Đương nhiên kết quả sẽ là 0 rồi.