# **Ý tưởng thuật toán**

# I. Thuật toán quy hoạch động

* 1. Bài toán đổi tiền.
     + Code: <https://ideone.com/xyPFAY>
     + Ý tưởng:

B1: Tạo 1 mảng đếm (có tên vt), trong đó vt[i] để lưu số lượng tờ tiền ít nhất mà i tiền đổi được, ban đầu khởi tạo tất cả các phần tử của vt đều là dương vô cùng, tạo mảng tien[] để lưu các mệnh giá tiền có sẵn.

B2: Duyệt lần lượt với biến chạy i từ 1 cho đến 10000 tương ứng với mỗi vòng for để tìm ra cách đổi i tiền ra số tờ ít nhất

B3: Tại mỗi lần đổi tiền, ta duyệt lần lượt các phần tử của mảng tien[]. Tại lần duyệt tien[j], nếu tờ tiền i đang cần đổi có mệnh giá lớn hơn tien[j] thì tiến hành đổi thử bằng cách kế thừa cách đổi mệnh giá (i-tien[j]) và thêm 1 tờ tiền - nghĩa là 1 + vt[i-tien[j]].

Nếu cách đổi thử có kết quả tốt hơn cách đổi hiện có, cập nhật kết quả đổi mới.

B4: In kết quả các lần truy vấn tương ứng để đổi m tiền thì in ra giá trị của vt[m].

* + - Mô tả tính tay:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test VT[i]  n Mệnh giá | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 2 | 0 | ++ | 1 | ++ | 2 | ++ | 3 | ++ | 4 | ++ | 5 |
| 3 | 0 | ++ | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 |
| 7 | 0 | ++ | 1 | 1 | 2 | 2 | 2 | 1 | 3 | 2 | 2 |

* 1. Sắp xếp ba lô
     + Code: <https://ideone.com/pCoVi2>
     + Ý tưởng: Khởi tạo 1 mảng dp[100][10000]. Trong bài toán có n đồ vật, đồ vật thứ i có kích thước và giá trị <wi;vi>. Gọi dp[i,j] là tổng giá trị lớn nhất của balo khi xét từ vật 1 đến vật i và kích thước balo chưa vượt quá j. Với giới hạn j, việc chọn tối ưu trong số các vật {1,2,…,i-1,i} để có giá trị lớn nhất sẽ có hai khả năng:

+ Nếu không chọn vật thứ i thì dp[i,j] là giá trị lớn nhất có thể chọn trong số các vật {1,2,…,i-1} với giới hạn kích thước là j, tức là:

dp[i, j] = dp[i-1, j]

+ Nếu có chọn vật thứ i (phải thỏa điều kiện W[i] ≤ j) thì dp[i, j] bằng giá trị vật thứ i là V[i] cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các vật {1,2,…,i-1} với giới hạn trọng lượng j - W[i] tức là về mặt giá trị thu được:

dp[i, j] = V[i]+dp[i-1,j-W[i]]

Vậy chúng ta phải xem xét xem nếu chọn vật i hay không chọn vật i thì sẽ tốt hơn. Từ đó chúng ta có công thức truy hồi như sau.

dp[0,j] = 0, dp[i, 0] = 0 (hiển nhiên)

dp[i,j]= max(F[i-1,j], V[i]+F[i-1,j-W[i]]

* + - Minh họa tính tay:

dp[2][4] = max(dp[1][4], dp[1][4-3]+5) = max(7,0+5) = 7

dp[2][5] = max(dp[1][5], dp[1][5-3] + 5) = max(7,0+5) = 7

dp[2][6] = max(dp[1][6], dp[1][6-3]+5) = max(7,0+5) = 7

dp[2][7] = max(dp[1][7], dp[1][7-3]+5) = max(7, 7 + 5) = 12

….

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n đồ vật | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ..... |
| (wi, vi) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| (4,7) | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 | 7 |  |
| (3,5) | 0 | 0 | 0 | 5 | 7 | 7 | 7 | 12 | 12 |  |
| (4,8) | 0 | 0 | 0 | 5 | 8 | 8 | 8 | 13 | 15 |  |
| (2,3) | 0 | 0 | 3 | 5 | 8 | 8 | 11 | 13 | 15 |  |
| (2,4) | 0 | 0 | 4 | 5 | 8 | 9 | 12 | 13 | 15 |  |
| (5,9) | 0 | 0 | 4 | 5 | 8 | 9 | 12 | 13 | 15 |  |

* M = 8 thì max = 15; M = 1 thì max = 0
  1. Bảng số.
     + Code: <https://ideone.com/RBu2n5>
     + Ý tưởng: gọi ma trận cần xét là (aij)n\*m (với 0≤i<n; 0≤j<m)

Dùng ma trận này để lưu đường đi lớn nhất từ góc trên trái tới vị trí bất kì.

Do chỉ có thể đi từ góc trên trái xuống góc dưới phải nên với các vị trí a[i][0] chỉ có thể đi từ a[i-1][0] xuống, còn các vị trí a[0][j] chỉ có thể đi từ a[0][j-1] sang.

Các vị trí còn lại thì có thể được đi từ ô ngay trên nó xuống, hoặc đi từ ô ngay bên trái sang, nên:

a[i][j] += max(a[i-1][j], a[i][j-1]);

Kết quả cần in ra là a[n-1][m-1], tương ứng với phần tử góc dưới-phải của ma trận.

* 1. Tìm tất cả những xâu con chung dài nhất. (chưa ok)
     + Code: https://ideone.com/FpxUXZ

# II. Quay lui

* 1. Bài toán Đổi tiền.
     + Code: <https://ideone.com/HCJl3Y>
     + Ý tưởng:

Gọi mảng tien[n] là mảng lưu mệnh giá tiền đang có sẵn trong ngân hàng.

Giả sử cần đổi số tiền bằng Test thì duyệt lần lượt mảng tien[n], nếu tại i bất kì mà Test>tien[i] thì đổi Test = 1 tờ tien[i] + đổi Test-tien[i]. Sau đó lại duyệt lần lượt mảng tien[n] để tìm số tờ nhỏ nhất đổi được của số tiền (Test-tien[i]), Có nghĩa là lúc này Test mới = Test – tien[i]. Tiếp tục làm cho tới khi test = 0 thì ta sẽ tìm được số tờ tiền đổi được.

|  |  |  |  |
| --- | --- | --- | --- |
| Test  Mệnh giá | 10 | 3 | 5 |
| 7 | 1+đổi 3 | ++ | ++ |
| 5 | 1+đổi 5 =1+1=2 | ++ | 1 |

* 1. Sắp xếp ba lô.
     + Code: <https://ideone.com/Fv2LRO>
     + Ý tưởng:

+ với test ba lô bất kì, khởi tạo 1 biến Res để lưu giá trị tối đa mà balo có thể đựng được. Quay lui duyệt lần lượt các đồ vật hiện có, tại mỗi đồ vật, ta được chọn lấy hoặc không lấy đồ vật hiện tại:

+ Nếu lấy đồ vật hiện tại, ta cập nhật Res nếu giá trị đồ vật hiện tại cộng với giá trị hiện có trong balo lớn hơn Res. Sau đó cập nhật kích thước trống mới cho balo bằng cách trừ đi khối lượng vật hiện tại, quay lui để bỏ vật tiếp theo vào balo sau khi đã bỏ vật hiện tại vào balo.

+ Nếu không lấy đồ vật hiện tại, trực tiếp xét đến đồ vật tiếp theo.

+ In ra Res khi xong hết tất cả quay lui.

* 1. Người đi du lịch.
     + Code: <https://ideone.com/fKSpZ2>
     + Ý tưởng:

Theo đề bài có n thành phố, đường đi qua tất cả các thành phố đúng 1 lần rồi trở về điểm xuất phát sẽ là hoán vị của dãy số từ 1 đến n.

* Sử dụng thuật toán quay lui, sinh hoán vị của dãy n phần tử, mỗi hoán vị là 1 lần đi thử qua tất cả các thành phố. Lưu dãy hoán vị đó vào 1 mảng hoanVi[16], quãng đường đi được tính bằng tổng quãng đường đi lần lượt từ thành phố đầu tiên của hoán vị cho đến thành phố cuối cùng của hoán vị cộng với khoảng cách của thành phố đầu và thành phố cuối. Quãng đường nhỏ nhất của hoán vị sẽ là đáp án cần tìm.
  1. Phân tích số (thành tổng một dãy không giảm các số nguyên dương).
     + Code: https://ideone.com/1mF62e
  2. Hoán vị lặp.
     + Code: <https://ideone.com/re62CD>
     + Ý tưởng: Gọi xâu đầu vào là sâu s. Tạo xâu tạm tmp ban đầu chưa có giá trị, duyệt lần lượt phần tử trong xâu s để đặt vào xâu tmp. Ta có 1 mảng ktra[14] ban đầu tất cả phần tử bằng 0 để đánh dấu xem kí tự thứ i của xâu s đã được thêm vào xâu tmp hay chưa.

Ta tiến hành duyệt lần lượt qua từng phần tử của xâu s. Nếu phần tử s[i] chưa được thêm vào, ta thêm phần tử s[i] vào cuối xâu tmp và đánh dấu phần tử thứ i đã được thêm: ktra[i]=1. Sau khi thêm phần tử s[i] vào xâu tmp, ta kiểm tra xem độ dài xâu tmp đã bằng độ dài xâu ban đầu hay chưa:

+) nếu đã bằng rồi thì ta đánh dấu mp[tmp] = 1; Sau đó xóa phần tử cuối cùng của xâu tmp đi, đánh dấu lại phần tử đó là chưa xét (ktra[i]=0) rồi lặp bước tiếp theo

+) nếu chưa bằng thì tiếp tục duyệt để thêm kí tự tiếp theo vào xâu tmp.

* + - Minh họa tính tay:
  1. Tám hậu.
     + Code: <https://ideone.com/QRQ59l>
     + Ý tưởng:

Sử dụng thuật toán quay lui và mảng đánh dấu, mục đích để đặt n quân hậu lần lượt lên n hàng ngang từ 1 đến n, mảng đánh dấu để đánh dấu các hàng chéo và hàng dọc mà các quân hậu được đặt trước đó đã chiếm. khi tìm dược ô trống trên hàng ngang đang xét mà không bị vướng bởi quân hậu nào (nhờ mảng đánh dấu kiểm tra), thì lập tức đặt quân hậu tại vị trí đó và xét các hàng ngang tiếp theo, khi xét xong trường hợp vừa rồi mới quay lại tìm vị trí thích hợp khác trên hàng ngang hiện tại.

Khi đặt được n quân hậu lên n hàng ngang thì tăng biến đếm kết quả thêm 1 đơn vị và in kết quả khi đã thử hết các trường hợp quay lui.

# III. Chia để trị

* 1. xâu Fibonacci.
     + Code: <https://ideone.com/YMhegD>
     + Ý tưởng

Vì xâu fibo thứ n bằng xâu fibo thứ n-2 ghép vs xâu thứ n-1 nên độ dài xâu thứ n bằng tổng độ dài xâu thứ n-2 cộng vs độ dài xâu thứ n-1. Với len(n1) = 1, len(n2) = 1.

* Độ dài của xâu fibo thứ i bằng số fibonanci thứ i.

Để tìm được kí tự thứ k trong xâu thứ n, ta xét

+ Nếu k <= len(n-2) thì kí tự cần tìm là kí tự thứ k của xâu n-2

+ Nếu k > len(n-2) thì kí tự cần tìm là kí tự thứ k-len(n-2) của xâu n-1

* Đệ quy để tìm kí tự cần tìm trong 2 xâu n-2 hoặc xâu n-1. Cho đến khi xâu n-2 hoặc xâu n-1 là 2 xâu cơ bản thì ta tìm được đáp án.
  1. Cắt thanh kim loại.
     + Code: <https://ideone.com/OPYuMv>
     + Ý tưởng:
       - dùng cây phân đoạn để quản lý thanh kim loại vs 1 node của cây sẽ có các thuộc tính:

+) Cận trái, Cận phải: Là cận của đoạn thanh kim loại mà node quản lý

+) Value: là độ dài đoạn dài nhất mà node quản lý

+) \*left, \*right: là 2 node con quản lý cây con trái và cây con phải của node.

Một node được khởi tạo với cận trái, cận phải biết trước, value là độ dài từ cận trái đến cận phải vì đoạn mới khởi tạo chắc chắn chưa bị cắt. (\*Left, \*Right đều bằng Null)

Cắt thanh tại 1 vị trí k nào đó, ta sẽ đi lần lượt từ node gốc cho đến node lá quản lý vị k và cắt đoạn đấy ra làm 2, tiếp theo update value của các node.

Value của 1 node bất kì sẽ là max của value 2 node con (nếu có)

* 1. Đếm số nghịch thế (cây IT).
     + Code: [h](https://ideone.com/rGjP9E)ttps://ideone.com/jt5Z2c
     + Ý tưởng:
  2. Đếm số nghịch thế (dựa mergesort).
     + Code: <https://ideone.com/lBBqZF>
     + Ý tưởng: Thực hiện theo thuật toán mergeSort: đặt left = 1, right = n, chọn m = (left+right)/2 để chia mảng cần sắp xếp thành 2 mảng con. (Gọi là mảng con b ở bên trái và mảng con c bên phải) Lại thực hiện chia đôi mảng con để sắp xếp, cứ như vậy cho tới khi mảng con chỉ có 1 phần tử thì thực hiện trộn 2 mảng con lại với nhau để được 1 mảng đã được sắp xếp. Trộn 2 mảng con b và c về mảng a ban đầu bằng cách:

1 biến i duyệt từ đầu mảng b và 1 biến j duyệt từ đầu mảng c, so sánh b[i] và c[j]:

+) nếu b[i]<c[j], viết b[i] vào mảng a sau đó tăng i lên 1 đơn vị

+) nếu c[j]<b[i] thì viết c[j] vào mảng a sau đó tăng j lên 1 đơn vị, trong trường hợp này, vì 2 mảng c và b đều đã được sắp xếp nên nếu c[j]<b[i] thì số nghịch thế sẽ được cộng thêm i.