Đồ án phân tích và thiết kế thuật toán

Subset-Sum Problem

1. Giới thiệu bài toán
2. Mô tả bài toán

Subset sum là một dạng Decision problem (Yes, No question). Được mô tả như sau: Cho một cặp (S, T) trong đó S là một mảng {x1,x2 …, xN} và T là một số nguyên. Có cách nào để bất kỳ mảng con nào của S tổng lại bằng chính xác T hay không

* Ví dụ: S = {-7, -3, -2, 9000, 5 , 8}

T = 0

Có cách nào để cộng mảng con S thành T = 0 không ?

Câu trả lời là có:

Mảng con {-3. -2, 5} cho ra tổng T = -3 + (-2) + 5 = 0

1. Ứng dụng

Subset sum được ứng dụng trong:

Password Encryption:

Máy tính sẽ tạo ra một con số lớn nào đó (Ví dụ: 500) gọi nó là A(i). Mật khẩu là một subset của mảng{1, . . ., 500}. Thay vì đưa người dùng mật khẩu, Máy tính sẽ giữ Tổng có liên quan tới subset dành riêng cho User. Khi User nhập Subset, máy tính sẽ test để xem Tổng có đúng không rồi cho User đăng nhập.

https://www.cs.princeton.edu/courses/archive/spring03/cs226/assignments/password

Message Verification:

Một người gửi (S) muốn gửi một tin nhắn của người nhận R. R muốn chắc chắn là tin nhắn là của S gửi, S và R đồng ý là cho mảng 500 A(i) và một mảng khác là (200) T(j). Hai mảng này có thể public nhưng chỉ có S mới biết mảng con nào của A[i] tương ứng với T[j]. Tin được gửi bởi S là một mảng con với kích thước là 100 của {1,2 . . , 200}và gửi 100 subsets của A[i] tương ứng với tin mà S muốn gửi

1. Đề bài với bài toán

Toàn là một người rất kín tiếng và có nhiều bí mật. Để bảo vệ bí mật của mình, Toàn đã nhờ Văn tạo ra một thuật toán để khởi tạo password để bảo vệ cái file quý của mình.

Thuật toán của văn rất đơn giản gồm 3 bước:

Bước 1: nhập 1 số nguyên N và M

Bước 2: máy tính sẽ tự tạo ra một mảng S có độ dài là N với các phần tử trong mảng có giá trị (0 < giá trị < N \* 2) được khởi tạo ngẫu nhiên không lặp

Bước 3: M sẽ là tổng mà mảng con của S phải tạo ra đề làm password

Input:

Hàng đầu tiên chứa 02 số nguyên dương N và M

Output: Mảng S và Mảng con bất kỳ làm password

Ví dụ: input N = 5, M = 6

Output: S = {1, 3, 5, 6, 9}

Sub = {1, 3}

1. Thiết kế thuật toán
2. Phương pháp: Backtracking
3. Lịch sử phát triển

**Quay lui** ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): *backtracking*) là một chiến lược tìm kiếm lời giải cho các [bài toán thỏa mãn ràng buộc](https://vi.wikipedia.org/w/index.php?title=B%C3%A0i_to%C3%A1n_th%E1%BB%8Fa_m%C3%A3n_r%C3%A0ng_bu%E1%BB%99c&action=edit&redlink=1). Người đầu tiên đề ra thuật ngữ này (*backtrack*) là [nhà toán học](https://vi.wikipedia.org/wiki/Danh_s%C3%A1ch_nh%C3%A0_to%C3%A1n_h%E1%BB%8Dc) người [Mỹ](https://vi.wikipedia.org/wiki/Hoa_K%E1%BB%B3) [D. H. Lehmer](https://vi.wikipedia.org/w/index.php?title=Derrick_Henry_Lehmer&action=edit&redlink=1) vào những năm 1950.

1. Pseudo code: #i = len(X)

Subset\_sum (X,i,S):

If S == 0:

Return true

Else if S < 0 or i = 0:

Return false

Else:

Return Subset\_sum(X,i-1,S) or Subset\_sum(X,i-1,S-X[i])

1. Phân tích độ phức tạp bằng phương pháp toán học

Time complexity: O(2\*N)

1. Mã nguồn cài đặt:

def subset\_sum(sett,i, summ):

    if summ == 0:

        return True

    elif summ < 0 or i == 0:

        return False

    else:

        return(subset\_sum(sett,i-1,summ) or subset\_sum(sett,i-1,summ-sett[i]))

a = list(map(int,input("\nEnter the numbers : ").strip().split()))

summ = int(input())

bruh = subset\_sum(a,len(a),summ)

if bruh == True:

    print('Correct')

else:

    print('False')

1. Cách thức phát sinh input/ output kiểm tra tính đúng đắn

Nếu input Sum = 0

Thì output = True vì Mảng con rỗng là kết quả

Hoặc nếu Sum = Số âm

Thì output = False

Hoặc Input X = [] và Sum != 0

Thì Output = False

Nếu X != [] và Sum != 0

Thì Output = True nếu tìm được mảng con

Output = False nếu không tìm được mảng con

1. Phân tích độ phức tạp = thực nghiệm
2. Phương pháp: Dynamic Programming
3. Trong ngành [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh), **quy hoạch động** (tiếng Anh: dynamic programming) là một phương pháp giảm thời gian chạy của các thuật toán thể hiện các tính chất của các [bài toán con gối nhau](https://vi.wikipedia.org/w/index.php?title=B%C3%A0i_to%C3%A1n_con_g%E1%BB%91i_nhau&action=edit&redlink=1) (overlapping subproblem) và [cấu trúc con tối ưu](https://vi.wikipedia.org/w/index.php?title=C%E1%BA%A5u_tr%C3%BAc_con_t%E1%BB%91i_%C6%B0u&action=edit&redlink=1) (optimal substructure).

Nhà toán học [Richard Bellman](https://vi.wikipedia.org/wiki/Richard_E._Bellman) đã phát minh phương pháp **quy hoạch động** vào năm [1953](https://vi.wikipedia.org/wiki/1953). Ngành này đã được thành lập như là một chủ đề về kỹ nghệ và phân tích hệ thống đã được tổ chức [IEEE](https://vi.wikipedia.org/wiki/IEEE) thừa nhận.

1. Pseudo Code:

#n = len(X)

Subset\_sum(X, n, Sum):

Khởi tạo M[0,r] = False, r = 0, . . ., Sum

Khởi tạo M[j,0] = False, j = 1,. . . , n

For i from 0, . . ., n:

M[i][0] = True

For i from 1, . . ., Sum:

M[0][r] = False

For i from 1, . . ., n:

For j from 1, . . ., Sum:

If j < X[i-1]:

M[i][j] = M[i-1][j]

If j >= X[i-1]:

M [i][j] = max(M[i-1][j],

M [i - 1][j-X [i-1]])

Return M[n][Sum]

1. Mã nguồn cài đặt
2. def subset\_sum(set, n, summ):
3. subset =([[False for i in range(summ + 1)]
4. for i in range(n + 1)])
5. for i in range(n + 1):
6. subset[i][0] = True
7. for i in range(1, summ + 1):
8. subset[0][i]= False
9. for i in range(1, n + 1):
10. for j in range(1, summ + 1):
11. if j<set[i-1]:
12. subset[i][j] = subset[i-1][j]
13. if j>= set[i-1]:
14. subset[i][j] = max(subset[i-1][j],
15. subset[i - 1][j-set[i-1]])
16. return subset[n][summ]
17. a = list(map(int,input("\nEnter the numbers : ").strip().split()))
18. summ = int(input())
19. bruh = subset\_sum(a,len(a),summ)
20. if bruh == True:
21. print('Correct')
22. else:
23. print('False')

5. Cách thức phát sinh Input/Output để kiểm tra tính đúng đắn

Nếu input Sum = 0

Thì output = True vì Mảng con rỗng là kết quả

Hoặc nếu Sum = Số âm

Thì output = False

Hoặc Input X = [] và Sum != 0

Thì Output = False

Nếu X != [] và Sum != 0

Thì Output = True nếu tìm được mảng con

Output = False nếu không tìm được mảng con

6. Phân tích độ phức tạp bằng thực nghiệm

Tài liệu tham khảo:

<http://www.math.stonybrook.edu/~scott/blair/Other_uses_subset_sum.html>

<https://vi.wikipedia.org/wiki/Quay_lui_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)>

<https://vi.wikipedia.org/wiki/Quy_ho%E1%BA%A1ch_%C4%91%E1%BB%99ng>