BÁO CÁO THỰC HÀNH MÔN PHÂN TÍCH THUẬT TOÁN

Lab 6 - Tuần 10

Đinh Anh Huy - 18110103

Bài toán. (Snapsack problem) Cho một tập hợp A gồm có n phần tử

$$A = \{a_1, a_2, ..., a_n\}, a_i \in \mathbb{R}(a_i > 0)$$

Với một số thực dương S, viết chương trình tìm một tập con $a_{i_1}, a_{i_2}, ..., a_{i_k}$ của A sao cho

$$a_{i_1} + a_{i_2} + ... + a_{i_k} = S \quad (i_j \neq i_l, \forall j, l \in \{1, 2, ..., k\})$$

- Input: $a_1, a_2, a_3, ..., a_n, S$.
- Output: $\{a_{i_1}, a_{i_2}, ..., a_{i_k}\}$.

Hint: Sử dụng phương pháp chia để trị hoặc quy hoạch động.
Test:

- $n = 50, 100, 150, ..., 950, 1000$.
- Tạo ngẫu nhiên $a_i \in \{0, 1, 2, ..., 500\}$.
- $S = 200$.
- Với mỗi n, tính thời gian trung bình để tìm một tập con có tổng bằng S. Giả sử thời gian trung bình là t.

Lời giải

Dùng phương pháp quy hoạch động để giải bài toán.
Ý tưởng:
Ta sẽ lập một bảng *True-False* có kích thước $(n + 1) \times (S + 1)$ chứa các giá trị $L(i, j)$ với $i = 1, ..., n, j = 1, ..., S$ là các giá trị kiểu *bool* thoả mãn

- $L[i, j] = True$ nếu có thể tạo ra tổng j từ một dãy con của dãy gồm các phần tử $a_1, a_2, ..., a_i$. Ngược lại thì $L[i, j] = False$.
- Nếu $L[n, S] = True$ thì tồn tại một tập con của A có tổng bằng S.
- Ta có thể tính $L[i, j]$ theo công thức:

$$L[i, j] = True \text{ nếu } L[i - 1, j] = True \text{ hoặc } L[i - 1, j - a[i]] = True$$

- Với $i = 0, 1, ..., n$, $L[i, 0] = True$ và với $j = 1, 2, ..., S$, $L[0, j] = False$.

Sau khi lập bảng chân trị như trên, nếu $L[n, S] = True$ thì ta tiến hành đi tìm các phần tử của tập con. Với các điều sau

- $L[i, j] = L[i - 1, j] = True$, phần tử $A[i]$ không nằm trong tập con.
- $L[i, j] = True$ và $L[i - 1, j] = False$, phần tử $A[i]$ phải nằm trong tập con, và đồng thời $L[i - 1, j - A[i]] = True$.

Từ các điều trên, ta có thể tìm các phần tử của tập con thoả mãn yêu cầu đề bài bằng đệ quy.

Cài đặt:

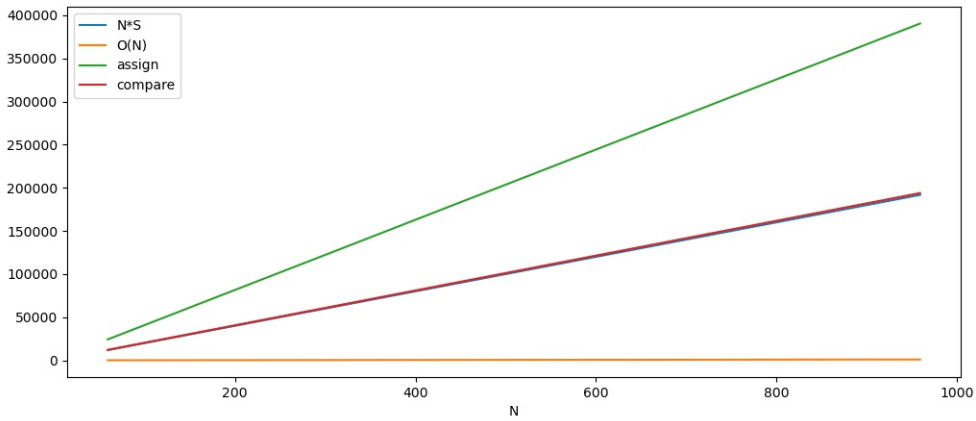
```
def isSubsetSum(S, M):
    n = len(S)
    # The value of subset[i, j] will be
    # true if there is a subset of
    # set[0..j-1] with sum equal to i
    subset = np.array([[True]*(M+1)]*(n+1))
    # If sum is 0, then answer is true
    for i in range(0, n+1):
        subset[i, 0] = True

    # If sum is not 0 and set is empty,
    # then answer is false
    for i in range(1, M+1):
        subset[0, i] = False

    # Fill the subset table in bottom-up manner
    for i in range(1, n+1):
        for j in range(1, M+1):
            if j < S[i-1]:
                subset[i, j] = subset[i-1, j]
            else:
                subset[i, j] = subset[i-1, j] or subset[i-1, j-S[i-1]]

    if subset[n, M]:
```

```
        sol = []
        # using backtracing to find the solution
        i = n
        while i >= 0:
            if (subset[i, M] and not subset[i-1, M]):
                sol.append(S[i-1])
                M -= S[i-1]
            if M == 0:
                break
            i -= 1
        return sol
    else:
        return None
```

Kết quả:

```
A =  [ 17  35  36  40  48  54  69  83  87  93  95 117 123 134 136 146 171 177
 181 185 199 221 227 231 236 242 253 275 281 284 296 340 347 364 369 370
 372 374 383 405 407 420 427 459 460 471 481 482 490 497]
S =  200
Found a subset with given sum.
Solution:  [69, 95, 36]
```

Ta dễ dàng thấy được thuật toán trên có độ phức tạp là $O(N \times S)$, khi S được cố định thì thuật toán có độ phức tạp là $O(N)$. Để kiểm chứng điều này ta vẽ đồ thị sau



Ta có thể thấy rằng đường thẳng $N \times S$ bị trùng với đường biểu diễn số phép so sánh của thuật toán. Do đó điều đó cho thấy rằng thuật toán thật sự có độ phức tạp là $O(N)$.