ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

PHÂN TÍCH THIẾT KẾ THUẬT TOÁN

Bài Toán Subset Sum

GV: Phạm Nguyễn Trường An

SV: Nguyễn Đức Thịnh - 18521442
Huỳnh Minh Tuấn - 18521596
Nguyễn Minh Thông - 18521459
Phan Phát Huy - 18520287

Mục lục

1 Giới Thiệu Bài Toán

1.1 Vấn đề

Bài toán Subset Sum là bài toán tìm kiếm trong tập hợp cho trước một tập hợp con có tổng bằng tổng cho trước.

Subset Sum được biết đến là bài toán NP-complete(Một bài toán L là NP-complete nếu nó nằm trong lớp NP (lời giải cho L có thể được kiểm chứng trong thời gian đa thức)).

1.2 Mô tả hình thức

Input: Set A có n phần tử là số nguyên dương và tổng cho trước Sum là một số nguyên không âm.

Output: Nếu tồn tại ít nhất một tập hợp con có tổng bằng Sum thì xuất ra $True$, ngược lại xuất ra $False$. Case nâng cao xuất ra sum maximum có thể được tạo ra nhưng không vượt quá Sum

1.3 Lịch sử

Là một vấn đề đã xuất hiện từ rất lâu trong quá trình nghiên cứu và phát triển tư duy của loài người trong lĩnh vực toán, tin học. Trong quá trình đó nhiều đề xuất để giải quyết vấn đề này đã được đề xuất như backtracking, bruteforce, và điển hình là dynamic programing được Bell Man đề xuất năm 1957.

1.4 Ứng dụng:

Có rất nhiều ứng dụng của Subset Sum trong đời sống, điển hình như Computer Password, Message Verification,...

1.4.1 Computer Password

Máy tính cần xác minh danh tính của người dùng trước khi cho phép họ truy cập vào tài khoản. Hệ thống đơn giản nhất sẽ yêu cầu máy giữ một bản sao của mật khẩu trong một tệp nội bộ và so sánh nó với những gì người dùng nhập. Một hạn chế là bất kỳ ai nhìn thấy tệp nội bộ sau đó đều có thể mạo danh người dùng. Máy tính trước tiên sẽ sản sinh ra 1 tập lớn các phần tử ai (chẳng hạn 500) và lưu trong local storage. Password là subset của 1,..500. Thay vì có mật khẩu cho người dùng, máy tính sẽ lưu tổng của subset. Khi người dùng nhập subset máy tính sẽ kiểm tra xem tổng có đúng hay không. Máy tính không lưu lại bản ghi nào của subset. Thế nên việc mạo danh chỉ xảy ra khi người khác biết tổng và subset.

1.5 Ứng dụng thực tế

Lazada là một nền tảng thương mại điện tử có số má ở Việt Nam. Vào mỗi dịp lễ Tết Lazada sẽ có những đợt giảm giá nếu mua đủ một khoảng n đồng nào đó. Tuy nhiêu chương trình chỉ cho phép mua mỗi mặt hàng với đơn vị là 1. Hãy viết chương trình giúp người dùng có thể biết được mình có húp được khuyến mãi này không.

Input: Tập hợp A với n phần tử thể hiện là số lượng hàng mà Lazada có trong mùa khuyến mãi mỗi phần tử có key là tên sản phẩm, value là giá tiền của sản phẩm đó, và giá tiền Sum phải

đạt để húp khuyến mãi.

Output: Nếu tồn tại ít nhất một tập hợp con có tổng bằng *Sum* thì thông báo khách hàng có thể đạt được khuyến mãi. Có thể trace back để tìm đó là những sản phẩm nào. Nếu không thể đạt được *Sum* thì báo cho khách hàng biết nên từ bỏ Lazada và chuyển sang nền tảng Shopee vì bên đó không có chơi khách hàng kiểu này.

2 Các Ý Tưởng Thiết Kế Thuật Toán:

2.1 Phát sinh test case

Phát sinh test case là một bước tất yếu của quá trình phát triển phân tích thiết kế một thuật toán.
Thay vì phải đau đầu nghĩ ra một test case nào đó cho bài toán đã có sẵn và đã được phân tích mòn từ nhiều thập kỷ qua nhóm nghĩ ngay tới việc sử dụng các test case có sẵn của các trang pratice code như hacker rank, leetcode, ...
Tuy nhiên các trang này thường không công bố test case giống như trang Wecode. Vậy nên nhóm sự dụng những test case có sẵn được công bố trong link sau https://people.sc.fsu.edu/~jburkardt/datasets/subset_sum/subset_sum.html.
Nếu thuật toán pass tất cả các test thì chứng tỏ được tính đúng đắn của nó

2.2 Phương pháp quay lui

2.2.1 Khái quát phương pháp

Backtraking là một kĩ thuật thiết kế giải thuật dựa trên đệ quy. Ý tưởng của Backtraking là tìm lời giải từng bước, mỗi bước chọn một trong số các lựa chọn khả dĩ và đệ quy. Người đầu tiên đề ra thuật ngữ này (backtrack) là nhà toán học người Mỹ D. H. Lehmer vào những năm 1950.

Ưu điểm:
Việc quay lui là thử tất cả các tổ hợp để tìm được một lời giải. Thế mạnh của phương pháp này là nhiều cài đặt tránh được việc phải thử nhiều trường hợp chưa hoàn chỉnh, nhờ đó giảm thời gian chạy

Nhược điểm:
Trong trường hợp xấu nhất độ phức tạp của quay lui vẫn là cấp số mũ. Vì nó mắc phải các nhược điểm sau:
- Rơi vào tình trạng "thrashing": qúa trình tìm kiếm cứ gặp phải bế tắc với cùng một nguyên nhân.
- Thực hiện các công việc dư thừa: Mỗi lần chúng ta quay lui, chúng ta cần phải đánh giá lại lời giải trong khi đôi lúc điều đó không cần thiết.
- Không sớm phát hiện được các khả năng bị bế tắc trong tương lai. Quay lui chuẩn, không có cơ chế nhìn về tương lai để nhận biết đc nhánh tìm kiếm sẽ đi vào bế tắc.

2.2.2 Mã giả

SubsetSum(X(1,2,....n), n ,T):
Input: T là tổng muốn đạt được
Input: X là tập hợp được cho
Input: n là Số phần tử của tập hợp X
if T = 0 :
return True
else if T < 0 or n == 0 :
return False
return **SubsetSum**(X[1,2,...,n-1],n-1, T) or **SubsetSum**(X[1,2,...,n-1],n-1,T-X[n])

2.2.3 Phân tích độ phức tạp bằng các phương pháp toán học

Phương trình đệ quy:

$$T(n) = \begin{cases} C_1, & n = 0 \\ 2T(n-1), & n > 0 \end{cases}$$

Giải phương trình đệ quy:

$$\begin{aligned} & T(n) = 2T(n-1) \\ \Leftrightarrow\ & T(n) = 4T(n-2) \\ \Leftrightarrow\ & T(n) = 8T(n-3) \\ \Leftrightarrow\ & T(n) = 16T(n-4) \end{aligned}$$

Sau i bước thay thế ta được:

$$T(n) = 2^i T(n-i)$$

Khi $n = i$:

$$\begin{aligned} & T(n) = 2^n T(0) \\ \Leftrightarrow\ & T(n) = 2^n C_1 \end{aligned}$$

Vậy độ phức tạp của thuật toán là : $O(2^n)$

2.2.4 Mã nguồn :

```
def isSubsetSum(set, n, sum):

    # Base Cases
    if (sum == 0):
        return True
    if (n == 0):
        return False

    # If last element is greater than
    # sum, then ignore it
    if (set[n - 1] > sum):
        return isSubsetSum(set, n - 1, sum)
```

```
13
14      # else, check if sum can be obtained
15      # by any of the following
16      # (a) including the last element
17      # (b) excluding the last element
18      return isSubsetSum(
19          set, n-1, sum) or isSubsetSum(
20          set, n-1, sum-set[n-1])
```

Listing 1: Solve by Backtracking

2.2.5 Phân tích độ phức tạp cài đặt bằng thực nghiệm

2.2.5.a Cấu hình máy thực nghiệm

- Chip: Intel Core i5 8250U
- RAM: 4GB
- Không card đồ hoạ

2.2.5.b Cách thức thực nghiệm

Ở mỗi lần gọi hàm *isSubsetSum* ta đều random 1 tập hợp *set* với số phần tử n tăng dần từ 20 đến 89. Đặt mốc thời gian trước và sau khi gọi hàm *isSubsetSum* để tính time diff và ghi kết quả vào cột *tn* tương ứng với mỗi n, quy trình này được thực hiện tự động bằng code.

Tiếp tục tính $y = f(n)$ với $f(n)$ là các hàm độ phức tạp thông dụng đã biết.

Sử dụng linear regrestion trên các giá trị tính được để tính hệ số X và MSE.

Ta có kết quả thực nghiệm như bên dưới.

2.2.5.c Kết quả thực nghiệm

n	tn	46.9693*lgn	20.6208*sqrtn	1.4961*n	0.211087*nlgn	0.0139272*n$\hat{2}$	0.00015241*n$\hat{3}$	1.40885E-25*2$\hat{n}$	3.47841E-135*n!
20	0.181176	202.9979	92.21902	29.922	18.24606	5.57088	1.21928	1.48E-19	8.45E-117
21	0.197245	206.3041	94.49638	31.4181	19.47038	6.141895	1.411469	2.95E-19	1.78E-115
22	0.227558	209.4564	96.72013	32.9142	20.70922	6.740765	1.622862	5.91E-19	3.90E-114
23	0.277083	212.4685	98.89388	34.4103	21.9619	7.367489	1.854372	1.18E-18	9.01E-113
24	0.311321	215.3525	101.0209	35.9064	23.22782	8.022067	2.106916	2.36E-18	2.16E-111
25	0.385314	218.1187	103.104	37.4025	24.50644	8.7045	2.381406	4.73E-18	5.39E-110
26	0.522199	220.7764	105.1459	38.8986	25.79724	9.414787	2.678758	9.45E-18	1.40E-108
27	0.632704	223.3337	107.1488	40.3947	27.09976	10.15293	2.999886	1.89E-17	3.79E-107
28	0.682402	225.7981	109.115	41.8908	28.41356	10.91892	3.345704	3.78E-17	1.06E-105
29	1.190402	228.176	111.0464	43.3869	29.73824	11.71278	3.717127	7.56E-17	3.07E-104
30	0.995859	230.4732	112.9448	44.883	31.07342	12.53448	4.11507	1.51E-16	9.22E-103
31	1.136592	232.6951	114.8118	46.3791	32.41876	13.38404	4.540446	3.03E-16	2.86E-101
32	1.279592	234.8465	116.6489	47.8752	33.77392	14.26145	4.994171	6.05E-16	9.15E-100
33	2.732696	236.9317	118.4575	49.3713	35.1386	15.16672	5.477158	1.21E-15	3.02E-98
34	1.997775	238.9546	120.2389	50.8674	36.51251	16.09984	5.990323	2.42E-15	1.03E-96
35	3.770094	240.9188	121.9943	52.3635	37.89537	17.06082	6.534579	4.84E-15	3.58E-95
36	2.65801	242.8278	123.7248	53.8596	39.28694	18.04965	7.110841	9.68E-15	1.29E-93
37	2.804384	244.6844	125.4314	55.3557	40.68697	19.06634	7.720024	1.93E-14	4.80E-92
38	3.465906	246.4915	127.1151	56.8518	42.09523	20.11088	8.363042	3.87E-14	1.82E-90
39	4.227073	248.2516	128.7769	58.3479	43.51151	21.18327	9.040809	7.75E-14	7.10E-89
40	4.564096	249.9672	130.4174	59.844	44.93559	22.28352	9.75424	1.55E-13	2.84E-87
41	5.584631	251.6405	132.0375	61.3401	46.36729	23.41162	10.50425	3.10E-13	1.17E-85
42	5.436054	253.2734	133.6381	62.8362	47.80642	24.56758	11.29175	6.20E-13	4.90E-84
43	6.539495	254.8679	135.2196	64.3323	49.2528	25.75139	12.11766	1.24E-12	2.10E-82
44	8.106527	256.4257	136.7829	65.8284	50.70626	26.96306	12.98289	2.48E-12	9.25E-81
45	8.26735	257.9485	138.3285	67.3245	52.16665	28.20258	13.88836	4.96E-12	4.17E-79
46	11.90808	259.4378	139.8571	68.8206	53.6338	29.46996	14.83498	9.92E-12	1.91E-77
47	9.980406	260.8952	141.3691	70.3167	55.10757	30.76518	15.82366	1.99E-11	9.01E-76
48	12.76195	262.3218	142.8651	71.8128	56.58782	32.08827	16.85533	3.96E-11	4.31E-74
49	13.26711	263.719	144.3456	73.3089	58.07442	33.43921	17.93088	7.93E-11	2.11E-72
50	11.83909	265.088	145.8111	74.805	59.56723	34.818	19.05125	1.59E-10	1.06E-70
51	18.03763	266.4298	147.262	76.3011	61.06614	36.22465	20.21734	3.17E-10	5.39E-69
52	16.08689	267.7457	148.6987	77.7972	62.57101	37.65915	21.43007	6.34E-10	2.81E-67
53	18.12508	269.0364	150.1217	79.2933	64.08175	39.1215	22.69034	1.27E-09	1.49E-65
54	22.18074	270.303	151.5313	80.7894	65.59822	40.61172	23.99909	2.54E-09	8.04E-64
55	21.47587	271.5464	152.9279	82.2855	67.12034	42.12978	25.35721	5.07E-09	4.42E-62
56	25.04538	272.7674	154.3119	83.7816	68.648	43.6757	26.76563	1.02E-08	2.47E-60
57	29.88423	273.9668	155.6836	85.2777	70.18109	45.24947	28.22527	2.03E-08	1.41E-58
58	28.4651	275.1453	157.0433	86.7738	71.71953	46.8511	29.73702	4.06E-08	8.17E-57
59	32.13041	276.3036	158.3914	88.2699	73.26322	48.48058	31.30181	8.11E-08	4.83E-55
60	32.11618	277.4425	159.728	89.766	74.81207	50.13792	32.92056	1.62E-07	2.89E-53
61	31.89916	278.5626	161.0536	91.2621	76.36599	51.82311	34.59417	3.25E-07	1.77E-51
62	41.21547	279.6644	162.3683	92.7582	77.92491	53.53616	36.32357	6.49E-07	1.10E-49
63	38.83752	280.7487	163.6725	94.2543	79.48874	55.27706	38.10966	1.30E-06	6.89E-48
64	42.40097	281.8158	164.9664	95.7504	81.05741	57.04581	39.95337	2.59E-06	4.42E-46
65	44.48521	282.8664	166.2502	97.2465	82.63083	58.84242	41.8556	5.20E-06	2.87E-44
66	52.45659	283.901	167.5242	98.7426	84.20894	60.66688	43.81727	1.04E-05	1.89E-42
67	58.10673	284.92	168.7885	100.2387	85.79166	62.5192	45.83929	2.09E-05	1.27E-40
68	52.98158	285.9239	170.0435	101.7348	87.37893	64.39937	47.92258	4.16E-05	8.63E-39
69	53.47965	286.9131	171.2892	103.2309	88.97068	66.3074	50.06806	8.31E-05	5.95E-37
70	52.46033	287.8881	172.526	104.727	90.56684	68.24328	52.27663	1.66E-04	4.17E-35
71	52.25237	288.8493	173.7539	106.2231	92.16735	70.20702	54.54922	3.32E-04	2.96E-33
72	55.56277	289.7971	174.9733	107.7192	93.77215	72.1986	56.88673	6.65E-04	2.13E-31
73	66.38579	290.7317	176.1842	109.2153	95.38118	74.21805	59.29008	1.33E-03	1.55E-29
74	65.29081	291.6537	177.3868	110.7114	96.99438	76.26535	61.76019	2.66E-03	1.15E-27
75	65.84824	292.5633	178.5814	112.2075	98.6117	78.3405	64.29797	5.33E-03	8.63E-26
76	76.25178	293.4608	179.768	113.7036	100.2331	80.44351	66.90433	1.07E-02	6.57E-24
77	67.78732	294.3466	180.9468	115.1997	101.8585	82.57437	69.58019	2.13E-02	5.04E-22
78	69.78137	295.2209	182.118	116.6958	103.4878	84.73308	72.32647	4.25E-02	3.93E-20
79	76.54159	296.0842	183.2817	118.1919	105.121	86.91966	75.14407	8.51E-02	3.11E-18
80	80.96433	296.9365	184.438	119.688	106.7581	89.13408	78.03392	1.70E-01	2.49E-16
81	87.53794	297.7783	185.5872	121.1841	108.3991	91.37636	80.99692	3.41E-01	2.02E-14
82	92.25078	298.6098	186.7293	122.6802	110.0437	93.64649	84.034	6.82E-01	1.65E-12
83	78.86256	299.4311	187.8644	124.1763	111.6921	95.94448	87.14606	1.36E+00	1.37E-10
84	86.03089	300.2427	188.9928	125.6724	113.3441	98.27032	90.33402	2.72E+00	1.15E-08
85	90.4318	301.0446	190.1144	127.1685	114.9998	100.624	93.59879	5.45E+00	9.81E-07
86	88.90865	301.8372	191.2294	128.6646	116.6591	103.0056	96.94129	1.09E+01	8.42E-05
87	82.86155	302.6205	192.338	130.1607	118.3219	105.415	100.3624	2.18E+01	7.34E-03
88	91.00732	303.395	193.4403	131.6568	119.9882	107.8522	103.8631	4.35E+01	6.44E-01
89	88.80662	304.1607	194.5362	133.1529	121.6579	110.3174	107.4443	8.72E+01	5.74E+01
MSE		54321.7	13696.9	2444.83	1263.8	226.597	30.8483	1.78E+03	1.98E+03

Từ thực nghiệm nhận thấy độ phức tạp của thuật toán là : $O(n) = n^3$

Tuy đã thực nghiệm lại nhưng vẫn không thu được kết quả 2^n

2.3 Phương pháp quy hoạch động

2.3.1 Khái quát phương pháp

Quy hoạch động là kĩ thuật được dùng khi có một công thức hoặc một (một vài) trạng thái bắt đầu. Một bài toán được tính bởi các bài toán nhỏ hơn đã được tìm ra từ trước, và kết quả các bài toán sẽ được lưu lại để những lần tính toán tiếp theo nếu cần đến những kết quả đó thì không cần tốn thêm thời gian thực hiện lại những bài toán này nữa. Nói cách khác, quy hoạch động bắt đầu từ việc giải các bài toán nhỏ, để từ đó từng bước giải quyết bài toán lớn hơn , và cuối cùng là bài toán lớn nhất (bài toán ban đầu).

Ý tưởng: Sử dụng hướng tiếp cận **bottom - up** để xây dựng mảng 2 chiều, ta tối ưu sum từ $1 \rightarrow$ n với các giá trị trong set.

Ưu điểm:
- Chương trình chạy nhanh, mang tính tối ưu hóa cao.

Nhược điểm:
- Sự kết hợp giữa các bài toán con chưa chắc cho ta bài toán lớn.
- Số lượng các bài toán con cần giải quyết có thể rất lớn.

2.3.2 Mã giả

```
public function SubsetSum(Set, n ,Sum):
    Input: Set là tập hợp cho trước
    Input: n là số lượng phần tử trong Set
    Input: Sum là Tổng cho trước
    Đặt Subset là mảng 2 chiều (n+1)(Sum+1) lưu các giá trị False
    //nếu Sum = 0 thì kết quả là True

    for i = 1 to n + 1 do:
        Subset[i][0] = True

    // nếu Sum != 0 và Set = [] thì kết quả là False

    for i = 2 to Sum + 1 do:
        Subset[0][i] = False

    for i = 2 to n + 1 do :
        for j = 2 to sum + 1 do:
            if Set[i] > j then :
                Subset[i][j] = Subset[i-1][j]
            else:
```

Subset[i][j] = Subset[i-1][j] or Subset[i-1][j-set[i]

return Subset[n][Sum]

Độ phức tạp của thuật toán là : $O(n) = n * sum$ trong đó sum là tổng muốn tìm subset và n là số phần tử của tập hợp

Độ phức tạp bộ nhớ là: $O(n) = n * sum$ trong đó $n * sum$ là kích thước của ma trận 2D .

2.3.3 Mã nguồn

```
def isSubsetSum(set, n, sum):

    # 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 =([[False for i in range(sum + 1)]
            for i in range(n + 1)])

    # If sum is 0, then answer is true
    for i in range(n + 1):
        subset[i][0] = True

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

    # Fill the subset table in botton up manner
    for i in range(1, n + 1):
        for j in range(1, sum + 1):
            if j<set[i-1]:
                subset[i][j] = subset[i-1][j]
            if j>= set[i-1]:
                subset[i][j] = (subset[i-1][j] or
                                subset[i - 1][j-set[i-1]])

    return subset[n][sum]
```

Listing 2: Solve Dynamic Programing

2.3.4 Trace Back

2.3.4.a Mã giả

traceBack(DP, n, sum, set):
Input: DP là matran dynamic programing nếu hàm isSubsetSum return về
Input: set là tập hợp được cho

Input: n là Số phần tử của tập hợp set
Input: sum là tổng muốn đạt được
Đặt m = n, b = sum
Khởi tạo mảng subset rỗng để lưu lại path
while b > 0 :
    if DP[m-1][b] == true do
        m = m - 1
    else :
        m = m - 1
        push set[m] into subset
        b = b - set[m]
print subset

2.3.4.b Mã nguồn

```
def traceBack(DP, sum, n, set):
    m, b = n, sum
    subset = []
    while b > 0:
        if DP[m-1][b]:
            m -= 1
        else:
            m -= 1
            subset.append(set[m])
            b -= set[m]
    print(subset)
```

Listing 3: Trace back Dynamic Programing

2.3.5 Phân tích độ phức tạp bằng phương pháp thực nghiệm

2.3.5.a Cấu hình máy thực nghiệm

- Chip: Intel Core i5 8250U
- RAM: 4GB
- Không card đồ hoạ

2.3.5.b Cách thức thực nghiệm

Ở mỗi lần gọi hàm *isSubsetSum* ta đều random 1 tập hợp *set* với số phần tử n tăng dần từ 20 đến 59. Đặt mốc thời gian trước và sau khi gọi hàm *isSubsetSum* để tính time diff và ghi kết quả vào cột *tn* tương ứng với mỗi n, quy trình này được thực hiện tự động bằng code.

Tiếp tục tính $y = f(n)$ với $f(n)$ là các hàm độ phức tạp thông dụng đã biết.

Sử dụng linear regrestion trên các giá trị tính được để tính hệ số X và MSE.

Ta có kết quả thực nghiệm như bên dưới.

2.3.5.c Kết quả thực nghiệm

n	tn	3.88060*lgn*sum	1.88234*sqrtn*sum	0.154339*n*sum	0.0230370*nlgn*sum	0.00193887*n2*sum	0.0000300791*n3*sum	8.76851E-18*2ñ*sum	2.31541E-80*n!*sum
20	4.030974	16.77167	8.41808	3.08678	1.991285	0.775548	0.240633	9.19E-12	5.63E-62
21	3.863134	17.04483	8.625966	3.241119	2.124902	0.855042	0.278563	1.84E-11	1.18E-60
22	3.65036	17.30527	8.828957	3.395458	2.260102	0.938413	0.320282	3.68E-11	2.59E-59
23	3.830771	17.55413	9.027386	3.549797	2.396814	1.025662	0.365972	7.36E-11	6.00E-58
24	4.05099	17.79241	9.221545	3.704136	2.534971	1.116789	0.415813	1.47E-10	1.44E-56
25	4.226884	18.02095	9.4117	3.858475	2.674513	1.211794	0.469986	2.94E-10	3.59E-55
26	4.663252	18.24053	9.598088	4.012814	2.815385	1.310676	0.52867	5.88E-10	9.33E-54
27	5.068824	18.45182	9.780926	4.167153	2.957535	1.413436	0.592047	1.18E-09	2.52E-52
28	5.266806	18.65542	9.960407	4.321492	3.100917	1.520074	0.660296	2.35E-09	7.06E-51
29	5.16485	18.85188	10.13671	4.475831	3.245486	1.63059	0.733599	4.71E-09	2.05E-49
30	5.007435	19.04168	10.31	4.63017	3.391201	1.744983	0.812136	9.42E-09	6.14E-48
31	4.937482	19.22525	10.48043	4.784509	3.538024	1.863254	0.896086	1.88E-08	1.90E-46
32	5.355774	19.403	10.64812	4.938848	3.68592	1.985403	0.985632	3.77E-08	6.09E-45
33	5.485246	19.57528	10.81322	5.093187	3.834854	2.111429	1.080953	7.53E-08	2.01E-43
34	5.485434	19.74241	10.97583	5.247526	3.984796	2.241334	1.182229	1.51E-07	6.83E-42
35	5.906561	19.9047	11.13607	5.401865	4.135715	2.375116	1.289641	3.01E-07	2.38E-40
36	6.290304	20.06241	11.29404	5.556204	4.287584	2.512776	1.40337	6.03E-07	8.61E-39
37	6.096659	20.2158	11.44983	5.710543	4.440377	2.654313	1.523597	1.20E-06	3.20E-37
38	6.182617	20.36511	11.60352	5.864882	4.594067	2.799728	1.6505	2.41E-06	1.21E-35
39	6.20609	20.51053	11.75521	6.019221	4.748633	2.949021	1.784262	4.82E-06	4.72E-34
40	6.280319	20.65227	11.90496	6.17356	4.90405	3.102192	1.925062	9.65E-06	1.89E-32
41	6.473785	20.79052	12.05286	6.327899	5.060299	3.25924	2.073082	1.93E-05	7.76E-31
42	6.819306	20.92543	12.19896	6.482238	5.217358	3.420167	2.2285	3.86E-05	3.26E-29
43	7.083658	21.05716	12.34333	6.636577	5.375209	3.584971	2.391499	7.72E-05	1.40E-27
44	7.121018	21.18587	12.48603	6.790916	5.533833	3.753652	2.562258	1.54E-04	6.16E-26
45	7.061773	21.31169	12.62712	6.945255	5.693212	3.926212	2.740958	3.09E-04	2.78E-24
46	7.325821	21.43473	12.76665	7.099594	5.85333	4.102649	2.927779	6.17E-04	1.27E-22
47	7.459951	21.55514	12.90467	7.253933	6.01417	4.282964	3.122902	1.24E-03	6.00E-21
48	7.588139	21.67301	13.04123	7.408272	6.175717	4.467156	3.326508	2.46E-03	2.87E-19
49	8.05135	21.78844	13.17638	7.562611	6.337957	4.655227	3.538776	4.94E-03	1.41E-17
50	9.170379	21.90155	13.31015	7.71695	6.500876	4.847175	3.759888	9.91E-03	7.04E-16
51	7.931418	22.01241	13.4426	7.871289	6.664459	5.043001	3.990023	1.97E-02	3.59E-14
52	7.960861	22.12113	13.57375	8.025628	6.828694	5.242704	4.229362	3.95E-02	1.87E-12
53	8.090724	22.22777	13.70364	8.179967	6.993567	5.446286	4.478086	7.90E-02	9.89E-11
54	9.425465	22.33242	13.83232	8.334306	7.159069	5.653745	4.736375	1.58E-01	5.35E-09
55	8.944139	22.43514	13.95981	8.488645	7.325185	5.865082	5.00441	3.16E-01	2.94E-07
56	8.962766	22.53602	14.08614	8.642984	7.491906	6.080296	5.282371	6.32E-01	1.65E-05
57	10.00026	22.63511	14.21136	8.797323	7.65922	6.299389	5.570439	1.26E+00	9.38E-04
58	9.966221	22.73248	14.33547	8.951662	7.827118	6.522359	5.868793	2.53E+00	5.44E-02
59	9.630325	22.82818	14.45853	9.106001	7.995588	6.749206	6.177615	5.05E+00	3.22E+00
MSE		189.819	26.6282	0.32812	2.89558	10.8037	18.0377	42.1988	4.49E+01

Từ thực nghiệm nhận thấy độ phức tạp của thuật toán là : $O(n) = n * sum$

2.4 Bonus Advance case: Nếu việc tìm subset tạo thành Sum là không thể thì tìm kết quả gần sum nhất mà không vượt qua nó

2.4.1 Giải quyết bằng Backtracking

2.4.1.a Mã giả

BackTracking(X, i, sum, k, n):
  Input: k là tổng muốn đạt được
  Input: X là tập hợp được cho
  Input: n là Số phần tử của tập hợp X
  Input: i bước quay lui
  Input: sum tổng tạo được ở bước thứ i
  if sum > k :
    return 0
  else if i == n :
    return sum
  return **Max**(**BackTracking**(i + 1, sum + set[i]), **BackTracking**(i + 1, sum))

2.4.1.b Mã nguồn

```python
def backtracking(i, sum):
    if sum > k:
        return 0
    if i == n:
        return sum
    pick = backtracking(i+1, sum + set[i])
    leave = backtracking(i+1, sum)
    return max(pick, leave)
```

Listing 4: Return nearest sum

2.4.1.c Thực nghiệm đánh giá độ phức tạp của case

Cấu hình máy thực nghiệm

- Chip: Intel Core i5 8250U
- RAM: 4GB
- Không card đồ hoạ

Cách thức thực nghiệm

Ở mỗi lần gọi hàm *backtracking* ta đều random 1 tập hợp *set* với số phần tử n tăng dần từ 20 đến 59. Đặt mốc thời gian trước và sau khi gọi hàm *backtracking* để tính time diff và ghi kết quả vào cột *tn* tương ứng với mỗi n, quy trình này được thực hiện tự động bằng code.

Tiếp tục tính $y = f(n)$ với $f(n)$ là các hàm độ phức tạp thông dụng đã biết.

Sử dụng linear regrestion trên các giá trị tính được để tính hệ số X và MSE.

Ta có kết quả thực nghiệm như bên dưới.

Kết quả thực nghiệm

n	tn	64.0656*lgn	32.1249*sqrtn	2.71991*n	0.411456*nlgn	0.0362706*n$\hat{2}$	0.000593185*n$\hat{3}$	2.64556E-16*2$\hat{n}$	8.37222E-79*n!
20	0.344618	276.8869	143.6669	54.3982	35.56566	14.50824	4.74548	2.77E-10	2.04E-60
21	0.44056	281.3965	147.2148	57.11811	37.95215	15.99533	5.493486	5.55E-10	4.28E-59
22	0.507078	285.6962	150.6791	59.83802	40.36692	17.55497	6.316234	1.11E-09	9.41E-58
23	0.622521	289.8047	154.0656	62.55793	42.80867	19.18715	7.217282	2.22E-09	2.16E-56
24	0.811223	293.7384	157.3792	65.27784	45.27625	20.89187	8.200189	4.44E-09	5.19E-55
25	0.865186	297.5114	160.6245	67.99775	47.76856	22.66913	9.268516	8.88E-09	1.30E-53
26	1.200351	301.1365	163.8055	70.71766	50.28463	24.51893	10.42582	1.78E-08	3.38E-52
27	1.306201	304.6247	166.9259	73.43757	52.82353	26.44127	11.67566	3.55E-08	9.12E-51
28	1.774107	307.9861	169.989	76.15748	55.38442	28.43615	13.0216	7.10E-08	2.55E-49
29	2.054637	311.2295	172.9979	78.87739	57.96652	30.50357	14.46719	1.42E-07	7.40E-48
30	2.334113	314.3629	175.9553	81.5973	60.56909	32.64354	16.016	2.84E-07	2.22E-46
31	2.932169	317.3936	178.8639	84.31721	63.19145	34.85605	17.67157	5.68E-07	6.88E-45
32	3.251418	320.328	181.7259	87.03712	65.83296	37.14109	19.43749	1.14E-06	2.20E-43
33	3.378786	323.1721	184.5435	89.75703	68.49303	39.49868	21.31729	2.27E-06	7.27E-42
34	4.857418	325.9314	187.3187	92.47694	71.17108	41.92881	23.31454	4.55E-06	2.47E-40
35	4.795174	328.6106	190.0535	95.19685	73.8666	44.43149	25.43281	9.09E-06	8.65E-39
36	6.298944	331.2143	192.7494	97.91676	76.57908	47.0067	27.67564	1.82E-05	3.11E-37
37	7.103429	333.7468	195.4081	100.6367	79.30805	49.65445	30.0466	3.64E-05	1.15E-35
38	7.723177	336.2116	198.0312	103.3566	82.05307	52.37475	32.54925	7.27E-05	4.38E-34
39	10.15684	338.6125	200.6199	106.0765	84.81371	55.16758	35.18714	0.000145	1.71E-32
40	11.99208	340.9525	203.1757	108.7964	87.58957	58.03296	37.96384	0.000291	6.83E-31
41	14.57703	343.2348	205.6997	111.5163	90.38027	60.97088	40.8829	0.000582	2.80E-29
42	14.24926	345.4621	208.1931	114.2362	93.18546	63.98134	43.94789	0.001164	1.18E-27
43	15.63854	347.6369	210.6571	116.9561	96.00478	67.06434	47.16236	0.002327	5.06E-26
44	19.0709	349.7618	213.0925	119.676	98.8379	70.21988	50.52987	0.004654	2.23E-24
45	21.77789	351.8389	215.5004	122.396	101.6845	73.44797	54.05398	0.009308	1.00E-22
46	26.01883	353.8703	217.8817	125.1159	104.5443	76.74859	57.73826	0.018616	4.61E-21
47	32.00005	355.8581	220.2372	127.8358	107.417	80.12176	61.58625	0.037233	2.17E-19
48	35.86215	357.804	222.5678	130.5557	110.3024	83.56746	65.60152	0.074466	1.04E-17
49	36.50591	359.7098	224.8743	133.2756	113.2001	87.08571	69.78762	0.148932	5.09E-16
50	53.18904	361.577	227.1573	135.9955	116.1099	90.6765	74.14813	0.297864	2.55E-14
51	51.08501	363.4073	229.4177	138.7154	119.0316	94.33983	78.68658	0.595727	1.30E-12
52	61.15088	365.2021	231.6559	141.4353	121.965	98.0757	83.40656	1.191454	6.75E-11
53	79.61625	366.9627	233.8728	144.1552	124.9097	101.8841	88.3116	2.382909	3.58E-09
54	75.44256	368.6903	236.0688	146.8751	127.8657	105.7651	93.40528	4.765817	1.93E-07
55	77.25711	370.3863	238.2446	149.5951	130.8326	109.7186	98.69115	9.531634	1.06E-05
56	89.28668	372.0517	240.4007	152.315	133.8104	113.7446	104.1728	19.06327	0.000595
57	101.5724	373.6876	242.5377	155.0349	136.7987	117.8432	109.8537	38.12654	0.03393
58	118.1911	375.2951	244.6559	157.7548	139.7975	122.0143	115.7375	76.25308	1.967942
59	140.0269	376.8751	246.756	160.4747	142.8065	126.258	121.8277	152.5062	116.1086
MSE		94782	29723.2	6581.42	3766.82	1301.28	425.758	141.24	1646.23

Từ thực nghiệm nhận thấy độ phức tạp của thuật toán là : $O(n) = 2^n$

Tài liệu

[1] Subset Sum problem
https://en.wikipedia.org/wiki/Subset_sum_problem

[2] Subset Sum problem
https://www.geeksforgeeks.org/subset-sum-problem-dp-25

[3] Paper
https://www.cs.dartmouth.edu/~deepc/Courses/S19/lecs/lec6.pdf

[4] Closest of Target
https://www.baeldung.com/cs/subset-of-numbers-closest-to-target

[5] Dataset (seeder)
https://people.sc.fsu.edu/~jburkardt/datasets/subset_sum/subset_sum.html