I.　その他頻出アルゴリズム

i.　bit 全探索

bit 全探索とは、選ぶ選ばないの 2 択を全ての組み合わせで試すアルゴリズムです。例えば、N 個の要素があるとき、それぞれの要素を選ぶか選ばないかの 2 択を 2^N 通り全て試すことができます。

部分和問題を使って bit 全探索の概要を説明します。部分和問題とは、N 個の整数 $a_1, a_2, \ldots, a_N$ が与えられたとき、それらの整数の中からいくつか選んで総和を K とすることができるかを判定する問題です。

$N = 3, K = 10, a = \{1, 4, 5\}$ のとき、a のそれぞれの要素を使うか使わないかの 2 択、全部で $2^3 = 8$ 通りの組み合わせがあります。それぞれの組み合わせに対して、総和が K となるかを判定します。

1	4	5	総和	判定
0	0	0	0	No
0	0	1	5	No
0	1	0	4	No
0	1	1	9	No
1	0	0	1	No
1	0	1	6	No
1	1	0	5	No
1	1	1	10	Yes

すべての場合を列挙してしまえば、あとは簡単に処理できそうですね。ではどのようにして列挙すればよいでしょうか。Python の標準ライブラリの itertools.product を使った方法や、bit 演算を使った方法があります。両方で部分和問題を解いてみましょう。

コード 1　itertools.product を使った bit 全探索

```python
from itertools import product

def bit_search(array: list[int], value: int) -> bool:
    size = len(array)
    flag = False

    for prod in product([0, 1], repeat=size):
        total = 0
        for index, to_use in enumerate(prod):
            if to_use:
                total += array[index]
```

```
12
13          if total == value:
14              flag = True
15
16      return flag
17
18  def main():
19      n, v = map(int, input().split())
20      A = list(map(int, input().split()))
21
22      is_ok = bit_search(A, v)
23
24      print("Yes" if is_ok else "No")
25
26
27  if __name__ == "__main__":
28      main()
```

itertools.product を使った実装は直感的に実装できましたが、この実装はどの言語でもできるわけではないので、bit 演算を使った実装を紹介します。bit 演算は 0 と 1 からなる 2 進数の数の各桁に対して、配列の要素をマッピングして 0 なら使わない、1 なら使うという処理を行います。$N = 3, K = 10, a = \{1, 4, 5\}$ のとき対応する 2 進数を考えると以下のようになります。0 と 1 の組み合わせをすべて列挙すればいいです。bit 演算という処理を使うことで、2^N 通りの組み合わせを簡単に列挙することができます。

1	4	5	総和	判定
0	0	0	0	No
0	0	1	5	No
0	1	0	4	No
0	1	1	9	No
1	0	0	1	No
1	0	1	6	No
1	1	0	5	No
1	1	1	10	Yes

bit 演算を理解するには、bit シフトや、AND や OR などの bit 演算子を理解する必要があります。bit 演算を使った実装は以下のようになります。`1 << n` は 2^n を表す bit シフトです。2^n 通りの組み合わせを列挙するために、2^N を計算しています。`bit & (1 << i)` は、`bit` の `i` 番目の bit

が立っているかどうかを判定しています。立っている場合は `A[i]` を総和に加えます。例えば、bit が 010010 であるとき、bit & $(1 < 2)$ とすれば 1 を 2 だけ左シフトした 100 と 010010 を AND するので、010010 に右から 3 番目の bit が立っているかどうかを判定できます。bit の位置と配列の要素の位置のマッピングを適切に行えば、部分和問題を解くことができます。下のコードでは bit の右の桁から順に A の元の要素に対応させています。

コード 2　bit 演算を使った bit 全探索

```
def main():
  n, v = map(int, input().split())
  A = list(map(int, input().split()))

  for bit in (1 << n):
      total = 0
      for i in range(n):
          if bit & (1 << i):
              total += A[i]

      if total == v:
          print("Yes")
          exit()

  print("No")


if __name__ == "__main__":
  main()
```

II.　3 つ以上の全列挙 (再帰による全列挙)

III.　問題

問題 1 AtCoder Beginner Contest 214 B How many?

問題 2 AtCoder Beginner Contest 367 C Enumerate Sequences

IV.　参考

bit 全探索

4 参考

- https://qiita.com/u2dayo/items/68e35815659b1041c3c2
- https://algo-method.com/tasks/1131I9eL