N의 배수

2020년 03월 15일

문제 설명

- $2N - 1$개의 수 중에서 합이 N의 배수가 되는 수 N개를 찾아라.

백트래킹 ($N \leq 10$)

- $N \leq 10$ 정도일 때 사용할 수 있는 풀입니다.
- 수 $2N-1$개 중, N개를 고르는 경우의 수는 $\binom{2N-1}{N}$입니다.
- 모든 가능한 경우를 따져서, 합이 N의 배수가 되는지 확인 해 줍니다.

동적 계획법($N \leq 500$)

- 수가 커졌기 때문에, 모든 경우를 확인할 수 는 없습니다.
- 우리는 동적계획법을 사용하여, 모든 경우를 탐색하지만 불필요하게 탐색하지는 않으려고 합니다.

동적 계획법($N \leq 500$)

- 현재까지 우리가 수 K개를 골랐고, 이 수들의 합을 N으로 나눈 나머지가 M이라고 합시다.
- 이 수들의 구성은 중요하지 않고, 골라야 하는 나머지 수들이 $N - K$개 이며, 합을 N으로 나눈 나머지가 $N - M$ 이어야 한다는 것을 알고 있습니다.
- "골라야 하는 나머지 수들"에서 고를 수 있는 수의 여부를 확실하기 위해, 수들을 앞에서 부터 확인한다고 합시다.

동적 계획법($N \leq 500$)

- 위의 성질을 사용하면, 다음과 같은 동적 계획법 테이블을 세울 수 있습니다.

동적 계획법 정의

- $D_{i,\ j,\ k} = i$ 번째 수 까지 확인 해 봤을 때, j개의 수를 골라 이들의 합을 N으로 나눈 나머지가 K가 되도록 할 수 있는가?

동적 계획법($N \leq 500$)

- 테이블은 다음과 같이 채워나갈 수 있습니다. (a_i를 i번째 수라고 합시다.)

동적 계획법 계산

- $D_{0,\ 0,\ 0} = true$.
  - 0개의 수 중 0개의 수를 골랐고, 합은 0입니다.
- $D_{i,\ j,\ k} = true \rightarrow D_{i+1,\ j,\ k} = true$.
  - $i + 1$ 번째 수를 고르지 않은 경우 입니다.
- $D_{i,\ j,\ k} = true \rightarrow D_{i,\ j+1,\ (k+a_{i+1}) \bmod N} = true$.
  - $i + 1$ 번째 수를 고른 경우에는 개수가 1, 합이 a_{i+1} 만큼 증가합니다.
- 위 규칙에 의해 *true*가 아니라면 모두 *false*입니다.

동적 계획법($N \leq 500$)

- 이제, 문제에서 원하는 $D_{2N-1,\ N,\ 0}$이 *true*인지 *false*인지 확인하면, 답이 가능한지 불가능 한지 여부를 확인할 수 있습니다.
- 이제 실제로 수 들을 출력하는 일이 남았습니다. 이는 거꾸로 i를 $2N-1$ 부터 보면서 a_i를 고르는 것이 가능한지 불가능 한지를 확인 해 봅니다.

동적 계획법($N \leq 500$)

- $D_{i,\ j,\ k}$가 *true*이고 $D_{i-1,\ j-1,\ (k-a_i) \bmod N}$가 *true*이면, j개의 합을 N으로 나눈 나머지가 k가 되도록 할 때, i번째 수를 사용하여 만들 수 있습니다.
- 만약 *false*라면, 우리는 $D_{i-1,\ j,\ k}$이 *true*임을 알 수 있습니다. (동적 계획법에 정의에 따릅니다.)
- 이 방식으로 $i = 2n - 1$ 부터 $i = 1$까지 어떤 수들을 사용했는지를 확인 해 줍니다.

동적 계획법($N \leq 500$)

- 한 가지 가능한 구현은, 현재 채워야 하는 집합의 j와 k값을 들고 다닙니다. 처음에 $j = N$, $k = 0$입니다.
- i를 $2n - 1$부터 1까지 감소시키면서, $D_{i-1,\ j-1,\ (k-a_i) \bmod N}$가 *true*이면, j를 $j - 1$로, k를 $(k - a_i) \bmod N$로 바꾸고, 집합에 a_i를 추가합니다.

Erdös-Ginzburg-Ziv theorem

- 수학적인 성질을 더 관찰하지 않고 시간 복잡도를 줄이는 것은 어려워 보입니다.
- 이제 수가 왜 하필 $2N - 1$개가 주어지며, 왜 아무리 데이터를 만들어 봐도 -1을 출력하는 데이터가 없는지 확인 해 봐야 할 때 입니다.
- 수가 $2N - 2$개가 주어지면 불가능한 경우가 있나요?

Erdös-Ginzburg-Ziv theorem

- 불가능한 경우는, 0이 $N-1$개, 1이 $N-1$개 있는 경우입니다. 여기서 총 N개를 고르면, 합은 1 이상 $N-1$이하가 되며, 이 중 어떤 수도 N의 배수가 아닙니다.
- 그럼 $2N-1$개가 주어지면, 항상 가능한가요?

Erdös-Ginzburg-Ziv theorem

- 답은 가능합니다. 이를 말해주는 정리는 Erdös-Ginzburg-Ziv theorem입니다.

Erdös-Ginzburg-Ziv theorem (1961)

- 수 $2n-1$개가 주어졌을 때 이 중 수 n개를 골라서 합이 n의 배수가 되게 할 수 있다.

Erdös-Ginzburg-Ziv theorem

- 이 증명은 N이 소수인 경우와 합성수 (혹은 1)인 경우로 나뉘어서 증명하며, 합성수에 대한 증명은, 깔끔하고 매력적인 결론 때문에 조합적 증명에 대한 연습문제로 자주 등장합니다.
- N이 소수인 경우의 증명이 어렵다고 알려져 있지만, 이 글에서는 간단한 증명을 소개합니다. 물론 이 증명을 직접 생각하는건 어렵다고 생각합니다.
- 우리는 이 증명에서 "강한 수학적 귀납법"을 사용할 것입니다.
- $N = 1$ 일 때 참이고, $N = 1, 2, \cdots, k-1$일 때 참인 결과로 $N = k$일 때를 증명할 수 있으면, 우리는 모든 N에 대해서 참이라는 것을 증명할 수 있습니다.

증명

- $N = 1$: 쉽습니다. 해당하는 수 하나를 골라주면, 그 수는 1의 배수입니다.
- $N \geq 2$, N은 합성수: $N = ab$라고 하고, $a, b \geq 2$라고 합시다. 우리는 $2a - 1$개를 골라서 합이 a의 배수가 되게 할 수 있고, $2b - 1$개를 골라서 합이 b의 배수가 되게 할 수 있다는 귀납 가정을 이용합니다. ($a, b < N$이기 때문에, 강한 수학적 귀납법을 사용합니다.)

Erdös-Ginzburg-Ziv theorem

N이 합성수 일 때 증명

- $N = ab$라고 하면, $2ab - 1$개의 수가 있습니다. 이 중 $2a - 1$개를 뽑아서, 합이 a의 배수가 되도록 할 수 있습니다.
- 이렇게 a개의 수를 제외하게 되면, 나머지는 $a(2b - 1) - 1$개의 수가 남게 됩니다.
- 위 작업을 x번 반복하게 되면, $a(2b - x) - 1$개의 수가 남게 됩니다. i번째 작업에서 뽑은 수들의 합을 s_i라고 합시다.
- 우리는 이 작업을 $a(2b - x) - 1 \geq 2a - 1$을 만족하면 할 수 있고, $x = 2b - 2$일 때 작업을 한 번 더해서 총 $2b - 1$번 작업을 할 수 있습니다.

Erdös-Ginzburg-Ziv theorem

N이 합성수 일 때 증명

- 이제 우리는 $\frac{s_1}{a}, \frac{s_2}{a}, \cdots, \frac{s_{2b-1}}{a}$이라는 새로운 $2b-1$개의 정수를 모았습니다. s_i는 모두 a의 배수이므로 a로 나누어도 정수입니다.
- 이 $\frac{s_i}{b}$들 중 b개를 뽑아서, 합이 b의 배수가 되도록 할 수 있습니다.
- 즉 s_i들 중에서 b개를 뽑아서, 합이 ab의 배수가 되도록 할 수 있습니다.
- s_i를 다시 a개의 수의 합으로 풀어 쓰면, 원래 $2N-1$개의 수 중 ab개의 수를 골라서 합이 ab의 배수가 되도록 할 수 있습니다.

Erdös-Ginzburg-Ziv theorem

- 이제 $N \geq 2$이고, N이 소수일 때 증명을 해야 합니다.

N이 소수 일 때 증명

- $2N - 1$개의 수 중에서 같은 수가 N개 이상 존재하면 해당 수를 N개 고른 경우에 합이 N의 배수가 됩니다.
- 같은 수가 N개 존재하지 않은 경우, 수 $2N - 1$개를 x, (a_1, b_1), (a_2, b_2), $\cdots$, (a_{N-1}, b_{N-1})과 같이 재배열 할 수 있습니다.
- 여기서 $a_i \neq b_i$를 만족하게 재배열 합니다. 또 편의상 $a_0 = b_0 = x$라고 합시다.

Erdös-Ginzburg-Ziv theorem

N이 소수 일 때 증명

- 우리는 S_k를 a_0, b_0 중 하나, a_1, b_1 중 하나, $\cdots$, a_k, b_k 중 하나를 고른 총 $k+1$개의 수의 합을 N으로 나눈 나머지의 집합 이라고 정의를 할 것입니다.
- 예를 들어, $S_0 = \{x\}$, $S_1 = \{x + a_1, x + b_1\}$, $S_2 = \{x + a_1 + a_2, x + a_1 + b_2, x + b_1 + a_2, x + b_1 + b_2\}$ 와 같은 방식으로 써내려 갑니다. (이후 증명에 나오는 덧셈에서 모든 합은 N으로 나눈 나머지입니다. 중복된 원소는 제거 되었습니다.)
- 이 때, 우리는 S_k의 원소의 개수는 $k+1$ 개 이상임을 수학적 귀납법으로 증명하겠습니다.

Erdös-Ginzburg-Ziv theorem

$|S_k| > k$

- $(k = 0)$: S_0 의 원소의 개수는 1개 이상임이 자명합니다.
- $(k = n \to k = n + 1)$: S_n의 원소의 개수가 $n + 1$개 이상이면, S_{n+1}의 원소의 개수는 $n + 1$ 개 이상임이 자명합니다.
- S_n의 원소의 개수가 n개 인 경우, 각 원소를 $\{s_1, s_2, \cdots, s_n\}$ 이라고 나열합시다.
- $A = \{a_{n+1} + s_1, a_{n+1} + s_2, \cdots, a_{n+1} + s_n\}$과 $B = \{b_{n+1} + s_1, b_{n+1} + s_2, \cdots, b_{n+1} + s_n\}$라고 하면, $S_{n+1} = A \cup B$입니다.

Erdös-Ginzburg-Ziv theorem

$|S_k| > k$

- $|A| = |B| = n$이기 때문에, $A \neq B$라면, $|A \cup B| \geq n+1$ 입니다.
- 우리는 $A \neq B$임을 보이기 위해서, A와 B에 있는 모든 수의 합을 n으로 나눈 나머지가 다르다는 것을 증명 합시다.
- $\sum_{a \in A} a - \sum_{b \in B} b = (na_{n+1} - \sum_{i=1}^{n} s_i) - (nb_{n+1} - \sum_{i=1}^{n} s_i) = n(a_{n+1} - b_{n+1})$입니다.
- $a_{n+1} \neq b_{n+1}$이고, $n < N$이기 때문에, $n(a_{n+1} - b_{n+1})$는 N의 배수가 될 수 없습니다.
- $A \neq B$이며, $|A \cup B| > n+1$입니다. $\square$

N이 소수 일 때 증명

- S_{N-1}의 원소의 개수는 $(N-1)+1=N$개 이상이기 때문에, 0부터 $N-1$까지의 수가 모두 포함되어 있으며, 0이 존재한다는 의미는, 합이 N의 배수가 되는 N개의 수가 존재한다는 의미입니다.

- 위와 같이 Erdös-Ginzburg-Ziv theorem이 증명되었습니다.

$N \le 5000$

- 위에 쓰인 증명을 구현해 주면 됩니다.
- $N = ab$가 합성수일 때는, 소인수 분해를 한 후, a와 b의 루틴을 사용하여 문제를 해결합니다.
- N이 소수일 경우에는, 증명에서 등장한 S_i 집합을 관리 해 줍니다. (역추적은 동적 계획법과 같은 방법으로 합니다.) S_i 하나를 관리 할 때 마다 $O(N)$ 시간이 들기 때문에, 총 $O(N^2)$ 시간이 듭니다.

$N \leq 5000$

- 시간 복잡도는 다음과 같습니다.
  - $N = ab$가 합성수: $T(ab) = (2b-1)T(a) + T(b) + O(N)$.
  - N이 소수: $T(N) = N^2$.
- 시간복잡도 분석을 통해, 모든 N에 대해 $T(N) = N^2$ 이라는 것을 알아낼 수 있습니다.
- $O((2b-1)a^2 + b^2 + O(ab)) < O(a^2b^2)$. (정확한 증명은 아닙니다.)

$N \leq 50000$

- $O(N^2)$풀이가 더 이상 동작하지 않습니다. 우리는 N이 소수인 경우에서, S_i가 0이상 $N-1$ 이하의 수를 담는다는 사실을 이용하여 'std::bitset' 등을 사용 해 최적화 해 줍니다.
- S_{i+1}을 구할 때는, S_i를 a_{i+1}번 shift한 벡터와 b_{i+1}번 shift한 벡터를 or연산 해 줍니다.
- 역추적은 $O(N^2)$풀이와 같은 방법으로 해 줍니다.

$N \leq 50000$

- 시간복잡도와 관련해서 주의해야 할 디테일 들이 있습니다. 예를 들면, $N = ab$를 소인수 분해 할 때, a를 작은 값으로 골라야 합니다.
- 구현의 편의등을 위해서 a를 큰 값으로 고르고, 수가 2^N꼴 일 때 시간복잡도 분석을 해 봅시다.
- $T(2^N) = 3T(2^{N-1}) + T(2) + O(2^N)$입니다.
- 매 단계마다 재귀호출로 호출되는 N의 합이 $\frac{3}{2}$배가 되고, $T(2^N) = O(3^N)$이 됩니다.
- 시간이 빡빡할 가능성이 높아 보입니다.

$N \leq 50000$

- 이를 모두 종합하면 문제를 해결할 수 있습니다!
- $N \leq 50000$인 경우에는 시간이 빡빡해서 생각보다 효율적인 구현이 필요합니다.
- 제가 구현한 코드에서는 S_i의 shift연산을 줄이기 위해서, 적당한 전처리를 통해 $b_{i+1} - a_{i+1}$ 만큼 한 번만 shift합니다.

모범 코드

- https://bit.ly/331HZ5B
- 해당 코드의 내용은 얼마든지 참고해도 좋지만, 푼 문제수를 늘리기 위해 그대로 복사해서 제출하는 것은 삼가 주시기 바랍니다.