**Main Theme**

크기가 S인 집합을 유한개의 크기 n을 가진 유한개의 숫자들의 부분 집합 k(n) 로 만들려 할 때, 각 집합의 평균 값이 다른 집합과의 평균 값이 가장 적은 집합의 형태의 해나 근사값을 구하라!

Ps. Research 결과 NP-완전 문제로 이 상황에 대한 보편 단일 해를 찾기 어려워, 근사값을 구하는 방식으로 전환합니다.

1st. think :

무작위의 k개의 집합을 생성

이후 각 집합의 평균을 구한 후, 평균의 편차가 가장 큰 2개의 집합에서 2개 집합의 집합 요소를 변형해 그 편차를 줄이는 방식으로 유도

Additional Think :

무작위 k개의 집합을 생성할 시 집합 요소들의 숫자를 내림/오름차순으로 정렬 후, 정렬된 숫자 중 index + n \* count

인덱스들을 집합의 요소로 n개의 인자로 구성된 k개의 부분 집합을 간단하게 축약된 집합을 만들어 이후 과정에서 계산을 줄입니다.

전제조건 :

집합 S을 크기가 n인 k개의 집합으로 나누고 이때 크기가 n인 집합이 전체 집합 S에서 만들 수 있는 최대 집합의 개수 k를 구해야 한다

고려 조건 :

스위칭 할 각 집합의 인자 Sa-k, Sb – k + 집합 S 내의 크기가 n인 k(n) 에 속하지 않는 외부 집합 O 등을 고려해야 함

**Propose**

평균의 편차가 가장 큰 두 집합 (… , a , ….) – Sa, (… , b , …..) - Sb 의 편차를 줄이는 조건에서

동일한 분모인 n을 가지는 평균값에서 각 부분집합의 총합의 차를 절대값으로 표현할 때,

그 값이 가장 적을 때 두 집합의 평균의 편차가 가장 적다고 할 수 있다.

위 식을 통해 Sa 와 Sb 의 임의의 두 요소를 교체하였을 때, 나타나는 식의 형태를 알 수 있다.

식의 구조를 단순화 하기 위해서 total of Sa > total of Sb 라는 전제에서

각 부분 집합의 평균을 줄이기 위한 임의의 두 요소를 선택하는 알고리즘을 구상해보자

두 집합의 편차를 줄이는 방식은 total of Sa 값을 줄이면서, total of Sb 값을 늘리는 방식으로 가능하다.

이 때, 위 조건들을 성립시키기 위해 (element of Sa : Sa-k) > (element of Sb : Sb-k) 라는 조건을 만족해야만 한다.

Sa 요소 값 교체로 인한 부분 집합의 값 감소 : {Sa -(Sa-k) + (Sb-k)} => -(Sa-k) + (Sb-k)< 0

* Sa의 요소 (Sa-k) 제거 후, Sb 의 요소 (Sb-k) 추가 -

Sb 요소 값 교체로 인한 부분 집합의 값 감소 : {Sb -(Sb-k) + (Sa-k)} => -(Sb-k) + (Sa-k) > 0

* Sa의 요소 (Sb-k) 제거 후, Sb 의 요소 (Sa-k) 추가 -

평균값의 편차에서 각 요소의 증감이 가능한 요소들을 Sa-k , Sb-k, 외부 집합 인자 O으로 가정하며

그 요소들의 존재 유무에 따른 각 집합의 변동폭은 이다.

| Mean of Sa – Mean of Sb | = B 라고 하자

이러한 조건 속에서 2개의 부분 집합의 평균 편차를 줄이는 방식은

Sthis 와 coupling 된 부분 집합 Sother 그리고 외부 부분 집합 O이 부분집합 Sthis 과 차이를 줄이는 방식은

위의 3가지 경우 중 R 의 값이 가장 적은 값이라고 할 수 있다.

**Prove of Deviation Reduction**

| Mean of Sa – Mean of Sb | = B

Mean of Sa 와 Mean of Sb 는 동일한 분모 n을 가진다

| Total of Sa - Total of Sb | = B \* N

동일 분모 n 을 가지고 있기 때문에 평균값 차이 B 에서 총합 차이 B \* N 이라고 할 수 있다.

이 경우 “B \* N 의 값이 줄어야 편차가 가장 적다” 고 할 수 있다..

If Sa > Sb 인 상황에서

Sa­-k <-> Sb-k swap 할 경우

(-a +b) - (a - b) = -2a +2b 의 값이 변동 값 이라고 볼 수 있다

이러한 결론으로

임을 알 수 있다

**INFINITE LOOP CHECK – Flicker Check**

Flicker 상태란? :

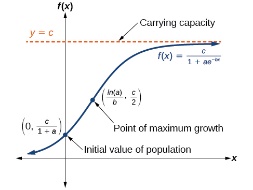
연산 과정 속에서 유한한 형태로 수렴되지 않고 “**반복되는**” 특정한 패턴의 형태를

연속적으로 보이는 상태 변화의 일종

* 수학에서의 발산과 비슷한 형태

Flicker 의 문제를 “**기계**” 단계에서 인지하는 작업이 필요

Case 1. Circle Loop Case 2. Infinite Limit Case

Sol. Pattern check Queue Sol. 임의의 값 EPSILON 설정

임의의 수 K 이하의 길이를 가진 패턴에 대해서 Epsilon 이하의 값이 나올 경우 break 통한 탈출

길이가 L인 check list 에서 매번 새로운 값의 인자가

들어 올 때, 인덱스 i 부터 L 까지 check pattern(cp)의

요소와 일치하는 케이스를 계산

Loop check

[ a, b, c e,f,g… ]

cp1,cp2,cp3

cp ----------------------------------------🡪