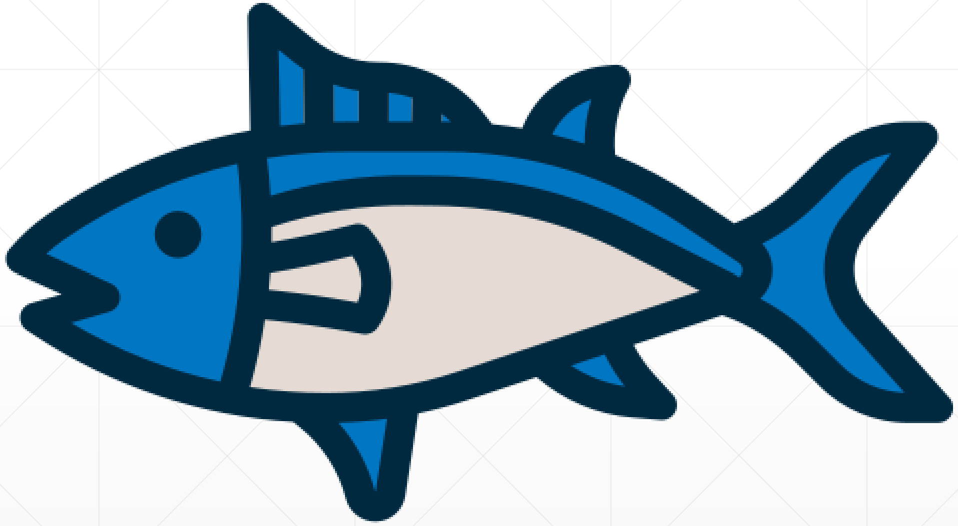
사조참치



박정환 범수 이신행 이창윤 윤태원

목차

문제 상황 설명 & 문제 단순화

첫 번째 아이디어 – **단순 조합 계산**

두 번째 아이디어 – **Naming bucket**

마지막 아이디어 – **Partition & Recursion**

문제 상황

고객들의 요구에 맞게 S&P 500 주식으로 이루어진 펀드 제작가능 여부 판단

조건1. S&P 500개의 주식만을 사용

조건2. 만들어진 펀드들에 동일한 주식은 없음

조건3. 펀드들에 있는 모든 주식의 합은 400개 이상

조건4. 펀드들의 개수는 몇 개이든지 상관없음.

다음 조건을 만족하는 **모든 경우**의 펀드를 구성하는 것이 현실적으로 가능할까??

문제 단순화 하기

서로 다른 공을 여러 바구니에 담는 모든 경우의 수를 계산할 수 있을까?

조건1. 500개의 서로 다른 공

조건2. 바구니 안에 들어있는 공들의 총 개수는 400개 이상

조건3. 바구니의 개수는 제한이 없음

첫 번째 아이디어

단순 조합 계산

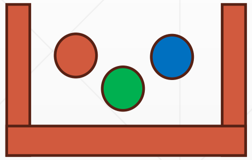
문제 최소화 하기

- 4개의 서로 다른 공을 여러 바구니에 담는 모든 경우의 수는?
- (단, 바구니에 담긴 공의 총 개수는 3개보다 많아야 한다.)

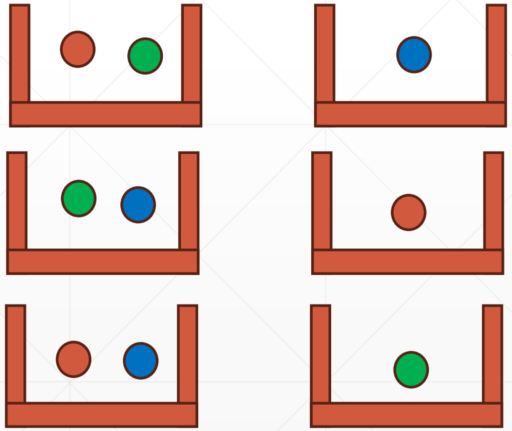
ANS = (공 **3**개를 바구니에 담는 경우의 수) + (공 **4**개를 바구니에 담는 경우의 수)

1. 서로 다른 공 **3개를 바구니에 담는 경우의 수**

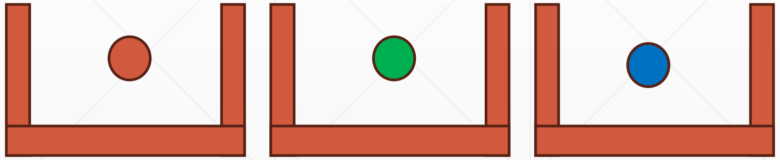
C(4,3) X (C(3, 3) + C(3, 2) X C(1, 1) + C(3, 1) X C(2, 1) X C(1, 1) / 3!   = 20가지



바구니 1개
C(3, 3) = 1개



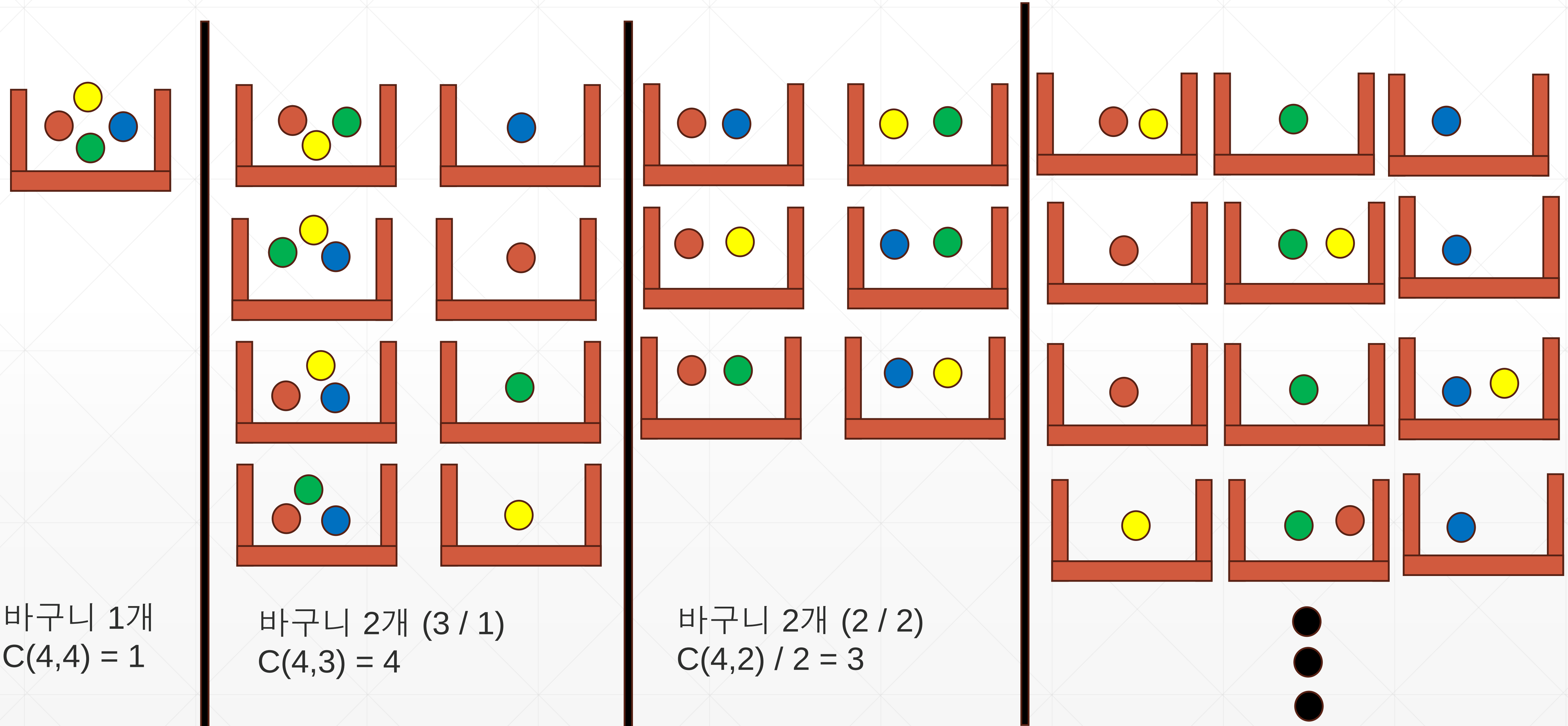
바구니 2개
C(3, 2) X C(1, 1) = 3개



바구니 3개
C(3, 1) X C(2, 1) X C(1, 1) / 3! = 1개

2. 서로 다른 공 4개를 바구니에 담는 경우의 수

${}_4C_4$ X $({}_4C_4 + {}_4C_3 * {}_1C_1 + ({}_4C_2 * {}_2C_2)/2! + ({}_4C_2 * {}_2C_1 * {}_1C_1)/2 + ({}_4C_2 * {}_2C_1 * {}_1C_1)/2 + ({}_4C_1 * {}_3C_1 * {}_2C_1 * {}_1C_1)/4!)$
= 15가지



500개로 확장하면 가능할까?

$$N = 400, \; , \; {}_{500}C_{400} \;\; X \; ({}_{400}C_{400} + {}_{400}C_{399} * {}_{1}C_{1} + ({}_{400}C_{398} * {}_{2}C_{1} * {}_{1}C_{1})/2!) + ({}_{400}C_{397} * {}_{3}C_{1} * {}_{2}C_{1} * {}_{1}C_{1})/3!) \ldots.$$

$$N = 401, \; , \; {}_{500}C_{401} \;\; X \; ({}_{401}C_{401} + {}_{401}C_{400} * {}_{1}C_{1} + ({}_{401}C_{399} * {}_{2}C_{1} * {}_{1}C_{1})/2!) + ({}_{401}C_{398} * {}_{3}C_{1} * {}_{2}C_{1} * {}_{1}C_{1})/3!) \ldots.$$

$$N = 402, \; , \; {}_{500}C_{402} \;\; X \; ({}_{402}C_{402} + {}_{402}C_{400} * {}_{1}C_{1} + ({}_{402}C_{399} * {}_{2}C_{1} * {}_{1}C_{1})/2!) + ({}_{402}C_{398} * {}_{3}C_{1} * {}_{2}C_{1} * {}_{1}C_{1})/3!) \ldots.$$

$$\vdots$$

에바 참치라는 결론에 도달

두 번째 아이디어
Naming Bucket

Naming Bucket 아이디어 설명

바구니(펀드)가 몇 개일 때를 기준으로 삼으면 되지 않을까?

뽑는 공(주식) : m과 바구니(펀드) : n 이라 가정
공(주식)m ≥ 400 이고 바구니(펀드) : n ≤ 공(주식) : m

1. 전체 공(주식)에서 바구니의 개수만큼 공을 뽑아 각 바구니에 1개씩 넣는다. : ${}_{500}C_{n}$
   1번은 모두 같은 바구니를 서로 다른 바구니로 구별하기 위한 과정

2. 남은 전체 공(주식) : 500-n에서 m-n개를 뽑는다. : ${}_{500-n}C_{m-n}$
   1번에서 n만큼 공을 뽑았기 때문에 남은 전체 공은 500-n
   총 n개의 공을 뽑기 때문에 m-n

3. 남은 공(주식)에게 바구니를 고르라고 한다 : n^{m-n}

공식으로 확장

따라서 일반항 : ${}_{500}C_{n}$ X ${}_{500\text{-}n}C_{m\text{-}n}$ X $n^{m\text{-}n}$

시그마를 이용한 최종공식 : $\sum_{m=400}^{500}\sum_{n=1}^{m} {}_{500}C_{n}$ x $500-nCm-n$ x n^{m-n}

코드 실험

```python
upper_bound = 500
lower_bound = 400
number  = 0
one_simul = 0

for i in range(lower_bound, upper_bound): #공의 개수 돌리기 EX) 400개부터 500개까지. 400,401,402,403
  select_number = len(list(combinations(list(range(1, upper_bound)), i))) # 몇개의 공을 할 지 정하기 ex) 400,
  for j in range(1,i): # 400개의 공을 가지고 1개부터 400개까지 바구니 개수 돌리기 이때 J는 바구니 개수, 1부터 400개까지 돌림
    one_simul += len(list(combinations(list(range(1, i+1)), j)))*math.pow(j,i)  #400개의 공에서 바구니 개수만큼 뽑아서 서로 다른 바구니를 만듬 =  len(list(combinations(list(range(1, i+1)), j)))
    # 서로다른 바구니 만들고 남은 공들에게 바구니 선택지를 제공 J의 I제곱
    # 위 두 값을 곱해서 바구니가 J일때의 경우의 수 한번을 구하고 one_simul에 더하기
  #안쪽 FOR문이 끝나고 나면 공 400개에 대한 모든 경우의 수를 구할 수 있음. 이 모든 값의 합을 one_simul에 저장한 상태
  number += one_simul * select_number  # 몇개의 공을 정할지 값에 one_simul을 곱함. 이를 400부터 500까지 반복함

print(number)

#오류 발생 , 런타임 RPM오류
```

하지만 코드를 돌려본 결과 : 런타임 오류(너무 방대한 크기의 데이터)

그래서 상황은 유지하되 숫자만 간략히 해서 다시 코드 실험을 진행

결과값이 다르게 나타남

값이 다르게 나온 이유

```
upper_bound = 4
lower_bound = 1
number   = 0
one_simul  = 0

for i in range(lower_bound, upper_bound): #공의 개수
  select_number = len(list(combinations(list(range(1,
  for j in range(1,i): # 400개의 공을 가지고 1개부터
    one_simul += len(list(combinations(list(range(1, i
    # 서로다른 바구니 만들고 남은 공들에게 바구니 선택
    # 위 두 값을 곱해서 바구니가 J일때의 경우의 수 한
  #안쪽 FOR문이 끝나고 나면 공 400개에 대한 모든 경우
  number += one_simul * select_number  # 몇개의 공을

print(number)

#오류 발생 , 런타임 RPM오류
# 이유분석 : 중복 제거 불가능.
```

```
35.0
```

예시 : 공 a,b,c,d를 2개의 바구니에 담는다.

총 경우의 수 : {(a),(b,c,d)} , {(b),(a,c,d)} , {(c),(a,b,d)} , {(d),(a,b,c)} , {(a,b),(c,d)} , {(a,c),(b,d)} , {(a,d),(b,c)}
7개

위 공식을 이용해 구한 경우의 수 : 24개

이유 :

a , b 와 c , d 는 다른 경우의 수

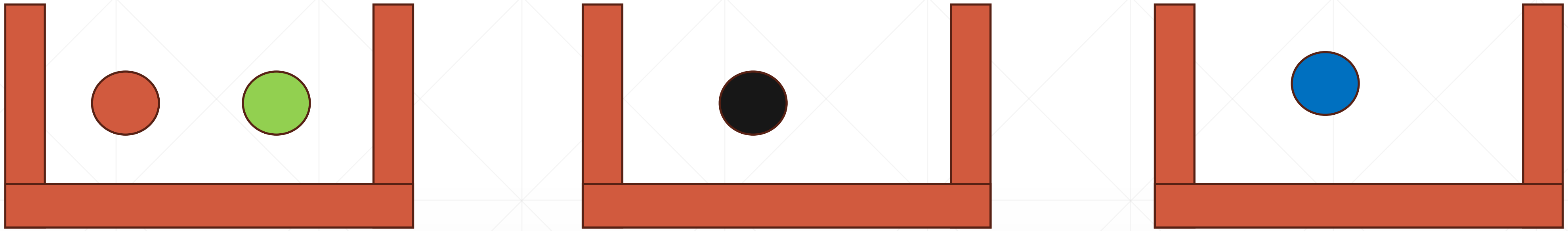
하지만 나머지 공을 무작위로 넣을 때

a,d , b,c 와 c,b , d,a 가 중복되는 경우 발생

중복되는 경우가 발생해 이 아이디어는 적합하지 않음

마지막 아이디어
Partition & Recursion

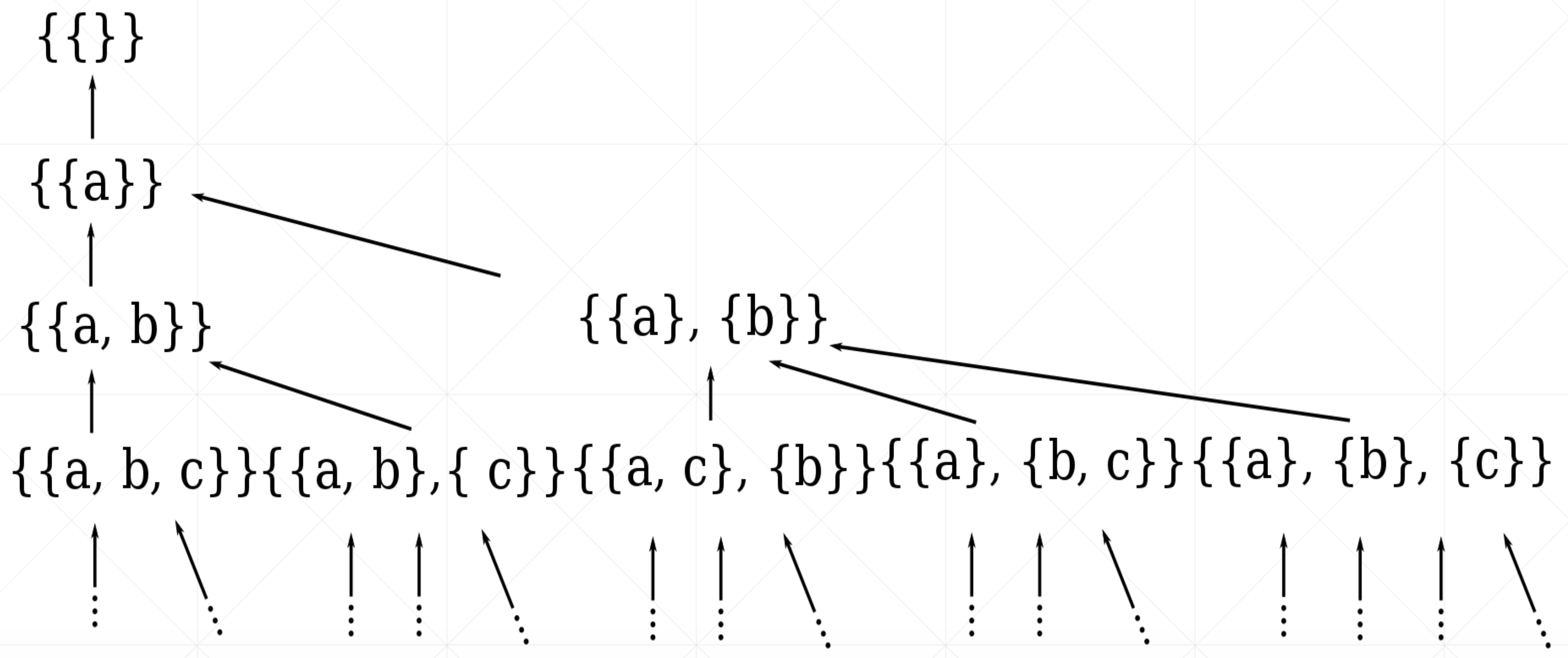
바구니 하나하나를 각 파티션이라고 볼 수 있지 않을까?



해당 문제 해결 아이디어

$$\sum_{k=400}^{500} \binom{500}{k} * \text{원소가 } k \text{일때의 파티션의 경우의 수의 총합}$$

Recursion



{a,b,c,d}	{a,b,d}{c}	{a,c,d}{b}	{b,c,d}{a}	{a,d}{b}{c}
{a,b,c}{d}	{a,b}{c,d}	{a,c}{b,d}	{a,d}{b,c}	{a}{b,d}{c}
	{a,b}{c}{d}	{a,c}{b}{d}	{b,c}{d}{a}	{a}{b}{c,d}
				{a}{b}{c}{d}

고등학생 때 배운 집합의 경우의 수 공식 : 제2종 스털링 수

$$S(n,k) = S(n-1,k-1) + kS(n-1,k)$$

S(n,k) :원소 개수가 N개인 집합을 k개의 부분집합으로 나누는 경우의 수

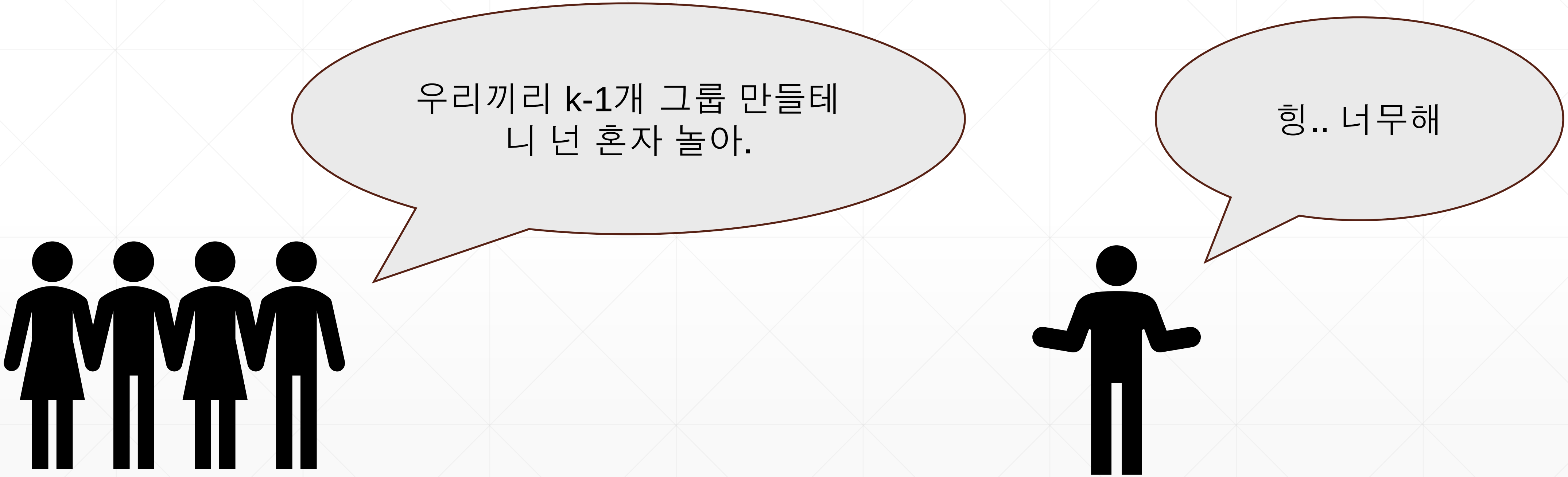
생활 예제. 고등학교 반에 전학생이 왔을 때 K개의 그룹을 만드는 방법은 뭘까요 ?

CASE 1. 전학생을 배제하고 자기들끼리 k-1개 그룹 만들기

CASE 2. 전학생이랑 함께 k개의 그룹 만들기

CASE 1. 전학생과 따로 놀기

기존 학생이 k-1개의 소그룹을 만들고, 전학생 혼자서 그룹을 만든다. → S(n−1,k−1)



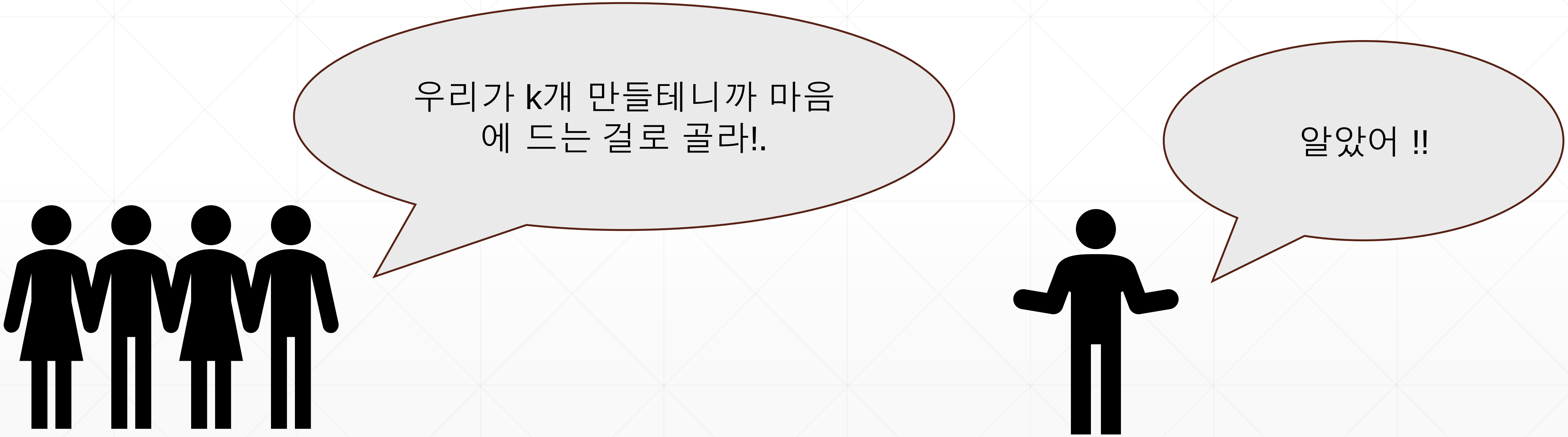
K-1개로 분할

혼자서 그룹

$$S(n,k) = S(n-1,k-1) + kS(n-1,k)$$

CASE 2. 전학생이랑 함께 놀기

기존 학생이 k개의 소그룹을 만들고, 전학생은 그 중 하나의 그룹에 들어간다 → S(n−1,k) * K



K-1개로 분할

혼자서 그룹

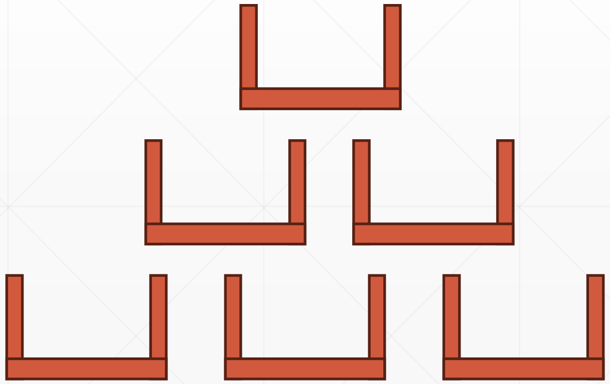
$$S(n,k) = S(n-1,k-1) + kS(n-1,k)$$

IDEA

$$\mathrm{Partition}(n) = \sum_{k=1}^{n} S(n,k)$$

IDEA

$$\mathrm{Partition}(n) = \sum_{k=1}^{n} S(n,k)$$



Bell number

Bell Number

$$B(n) = \sum_{k=1}^{n} S(n,k) = \sum_{k=0}^{n} \binom{n}{k} B(k)$$

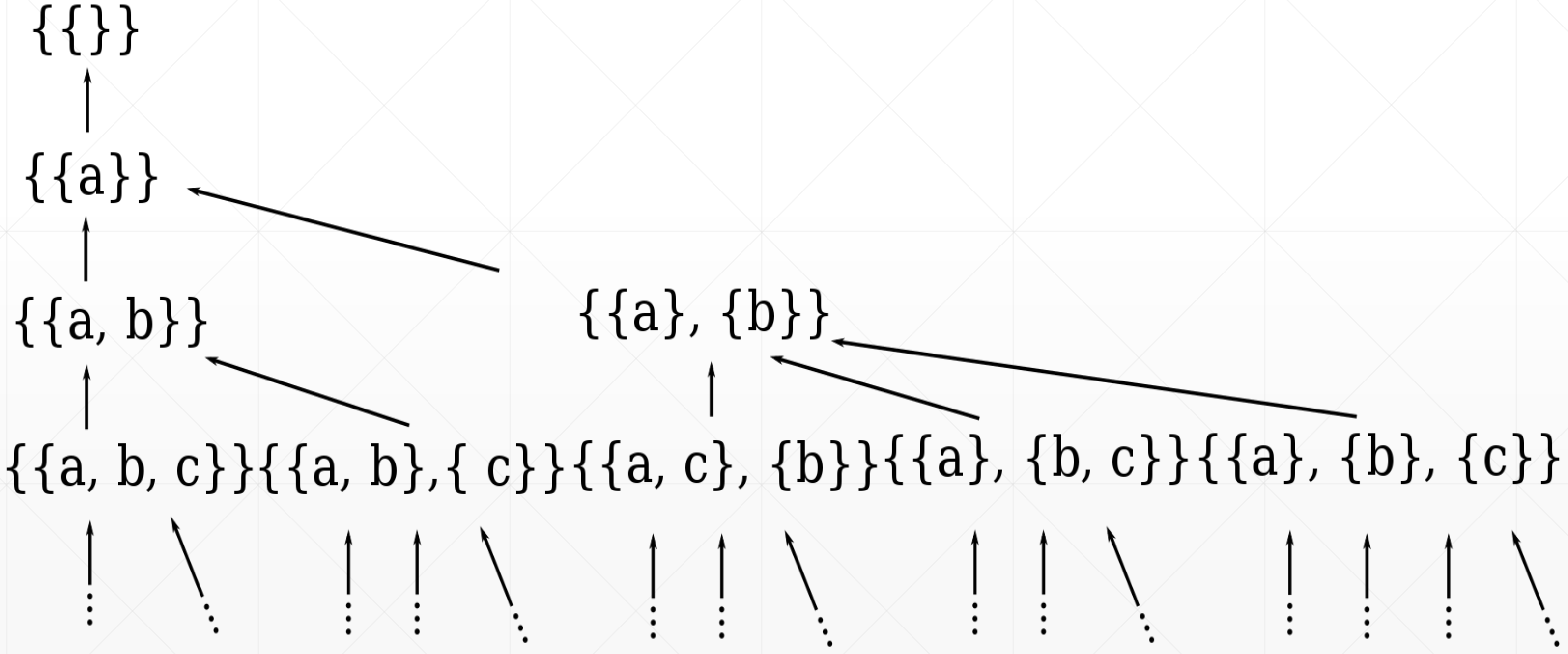
$B(0) = 1$

$B(n)$ is number of partitions of {1,2,3,4,…,n} into nonempty subsets

Bell Number

$\mathrm{B}(n)$ is number of partitions of {1,2,3,4,…,n} into nonempty subsets

If n = 3, $\mathrm{B}(3) = 5$



$\mathrm{B}(0) = 1$
$\mathrm{B}(1) = 1$
$\mathrm{B}(2) = 2$
$\mathrm{B}(3) = 5$
$\mathrm{B}(4) = 15$

Bell Number

$$B(n+1) = \sum_{k=0}^{n} \binom{n}{k} B(k)$$

$$= +\binom{n}{0} B(n)$$

$$+\binom{n}{1} B(n-1)$$

$$+\binom{n}{2} B(n-2)$$

$$+\binom{n}{3} B(n-3)$$

$$\cdots$$

$$+\binom{n}{n} B(0)$$

{1,2,3,4,…,n}

Subset with 1 element

Subset with 2 element

Subset with 3 element

.

.

.

Bell Number

$$B(n+1) = \sum_{k=0}^{n} \binom{n}{k} B(k)$$

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

$$B(x) = e^{e^x - 1}$$

$$B_{n+1} = \sum_{k=1}^{n+1} \binom{n}{k-1} B_{n-k+1} = \sum_{k=0}^{n} \binom{n}{k} B_{n-k} = \sum_{k=0}^{n} \binom{n}{n-k} B_k = \sum_{k=0}^{n} \binom{n}{k} B_k$$

Bell Number

$$B(x) = e^{e^x - 1}$$

$$= 1 + x + x^2 + \frac{5}{6}x^3 + \frac{5}{8}x^4 + \cdots$$

$$= 1 + \frac{1}{1!}x + \frac{2}{2!}x^2 + \frac{5}{3!}x^3 + \frac{15}{4!}x^4 + ..$$

$$= 1 + \frac{B(1)}{1!}x + \frac{B(2)}{2!}x^2 + \frac{B(3)}{3!}x^3 + \frac{B(3)}{4!}x^4 + \ldots$$

$$= \sum_{k=0}^{\infty} B(K)\frac{x^k}{k!}$$

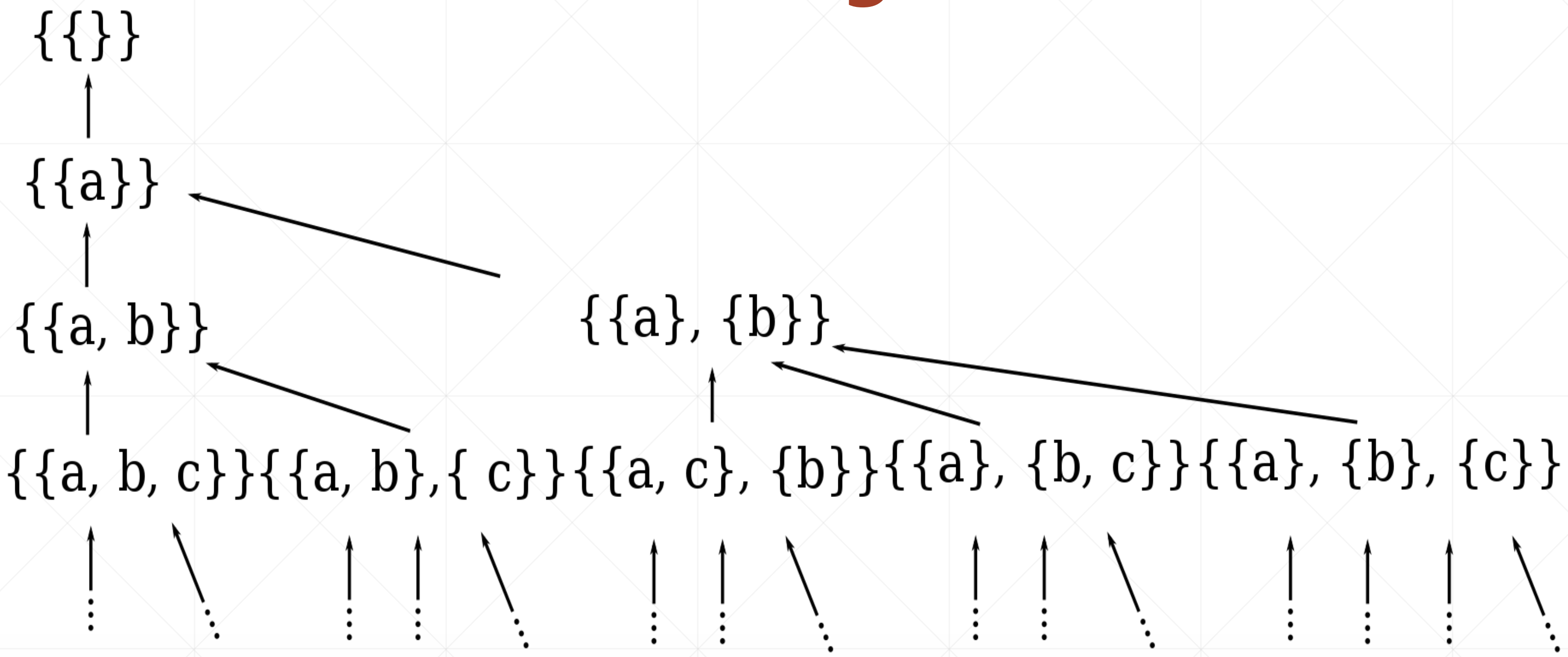
$B(1) = 1$
$B(2) = 2$
$B(3) = 5$
$B(4) = 15$

문제의 총 경우의 수 공식

$$\sum_{k=400}^{500} \binom{500}{k} B(k)$$

Bell Triangle

Bell Triangle



{a,b,c,d}	{a,b,d}{c}	{a,c,d}{b}	{b,c,d}{a}	{a,d}{b}{c}
{a,b,c}{d}	{a,b}{c,d}	{a,c}{b,d}	{a,d}{b,c}	{a}{b,d}{c}
	{a,b}{c}{d}	{a,c}{b}{d}	{b,c}{d}{a}	{a}{b}{c,d}
				{a}{b}{c}{d}

Bell Triangle

1

$B(0) = 1$
$B(1) = 1$
$B(2) = 2$
$B(3) = 5$
$B(4) = 15$

Bell Triangle code

```
def bellNumber(n):

    bell = [[0 for i in range(n+1)] for j in range(n+1)] #벨 삼각형 틀 만들기
    bell[0][0] = 1 #초기값 설정 b(1) = 1
    for i in range(1, n+1):

        # 새로운 행으로 내려갈 경우 이전 행의 맨끝값으로 정의
        bell[i][0] = bell[i-1][i-1]

        # 한칸 씩 이동하하면서 이전값과 이전행의 동일 위치에 있는 값을 더함
        for j in range(1, i+1):
            bell[i][j] = bell[i-1][j-1] + bell[i][j-1]

    return bell[n][0] # 출력
```

문제의 모든 경우의 수 개수세보기

$$\sum_{k=400}^{500} \binom{500}{k} * \text{원소가 } k \text{일때의 파티션의 경우의 수의 총합}$$

실험 진행

```python
# 실험 진행
for n in range(400,500):
    print('Bell Number', n, 'is', bellNumber(n))

# This code is contributed by Soumen Ghosh
```

```
Bell Number 400 is 12826199524676938087884998967316284449574181242767827088296318063624271385572606475000414936047907196150322022037441208409798299875412907
Bell Number 401 is 11436842649416682000146851600035900924454433665571080746858128638722620576584395693297768927520124058680378933603693800602293821761432862
Bell Number 402 is 10218810350232225148294337953664084403377736998572089631997889328321781394870309547457435164402017116512759329591412221581779537493910249
Bell Number 403 is 91491013658217798292195166861878360167534260140616538155105121477181215472678121094607662660588937303374981687747874156478543982783976940
Bell Number 404 is 82080184686424178353132489944060503712820779873013201563667559651546667934548097046846259241171085300390342941127343977546515855981645391
Bell Number 405 is 73786665587323267744535290248737347185603128013584318045013367525887661188990766899863981181319858351234231650043813795671476392990112011
Bell Number 406 is 66465307554229264297290317970446826239707445230563555137803500304036857673563372961573491808943739078504372757321324607422069325289813805
Bell Number 407 is 59991211867963298399225838641649326541392549681043969488309665954609432713892577923066197547911496619800953506663391086852036555266639604
Bell Number 408 is 54256734728500748627422405863778464911184819202551874078573626460081566854067972175108588296690620686375498702650561164002004598143186935
Bell Number 409 is 49168957843179828872672659220815566379626769744641264221173101524436406131879363629569601969718684527803474944735363828999399177147504741
Bell Number 410 is 44647549105289214917909701817613123560829004498457399416727053161673212373201832462567664099055900974145463422613456049953586243325365412
Bell Number 411 is 40622950478026999499870983862522212057790731579590050341095654109478413070532610971123417601216612221997625360504621646246105709839656851 0
Bell Number 412 is 37034840743851457741839414257241973073890513947556687753578523166080897424291802017883689442539972388949184175548386557742908740724416576
Bell Number 413 is 33830829492083821687081329437984254052213858027299714471299308336517336227539602510638342386763099134897073868858207196508295946228968166
```

실험 진행

400일때 숫자의 길이는 645
500일때 숫자의 길이는 844

실험 진행

```python
for i in range(400,501):
  ans = math.comb(500,i)
  print(ans)
```

```
204169423593847671561387240724193094030165460090325932474143661505758330944415515994945349100113181687417345
50915068227892187421792329357654138162136024960181030542180464215899833153220826931407817730701541567934500
12538785459107777499396618423899899696645438982731149312626532232273839507882740960719835709799633371207750
3049133932984025297620021353702705137149511216644299336569231163183216555266770754715989825211821514586500
732094038364976370963222948785055441345303435679448108037661937694980212526922681206561913479075957710125
173533401686512917561652847119420549059627481049943255238556903749921235561937228137851712824669856642400
40605106305957456079697094769322542267646824383607411940056418365129353148729154367231312114146887638000
9378083520294842436097117710850906567957743223732424379275929548703093847618035652382661765920902796000
2137651390655442025875078301738074291225662058350773204099660411836734627030581656057812608408441078500
480840899609537081614932038532769767219464326083792505567649774789681138598566044883419951035639558000
106723224059677742504777598796297680041393301642988092699161291477709716127974414839978574498154145800
23370049064163009307615532583130878841181014958318560445071815652053222509775419308024505364559302000
5048384385219679195091704854122932565206578474005708445658717458817322338276728928189759653994606500
1075684808472958278857312414437816139801885970248189693021712194614828972804726744989585592134444000
226049706128375290484507681294903391697497776356503631142243866984275653705341127570275233129702000
46844035486844036100404001424967931773457370522070631995742102555777605346167077038659445901576800
9571497635533036222438317598851620674865087726865393556822304608752635707750484491072242552004875
1928071466150539670706999228545650207886492491742669205690823949964559710913766660072106413353500
382846726532284192987274966433705663288466212475219004957747339347029799057039791354030699302250
74924657698442252565528752380820678734258304112095366125382534192020151605434994966659945925500
14449755413271005851923402244872559470178387221618392038466631594175314952476749029284418142775
2745796753115630565686157196175308212860501134749338154577982250674644171492018817916278982000
514023562787049323908072081748458172549714667405681787231423217543357558170306840320819999000
94784486755058740578793433513900088555266534415232102610049671320051748315092041477598014000
17213220472027176944733713161722421742347931957483188445692982763311284481750205645695865750
3078128837350742230117087530096244829219865479455817227935686329439194401442389715465613640
541924091082877153189628086284550145989412936523911483791494072084365211521547485117185500
93916587213426017180403930644160915230015356680958898830376021860053924244952023181901000
16018483333131073023760483497719034606988600555397195361255723354635365583835274981959750
```

$$\sum_{k=400}^{500} \binom{500}{k} * \text{원소가 } k\text{일때의 파티션의 경우의 수의 총합}$$

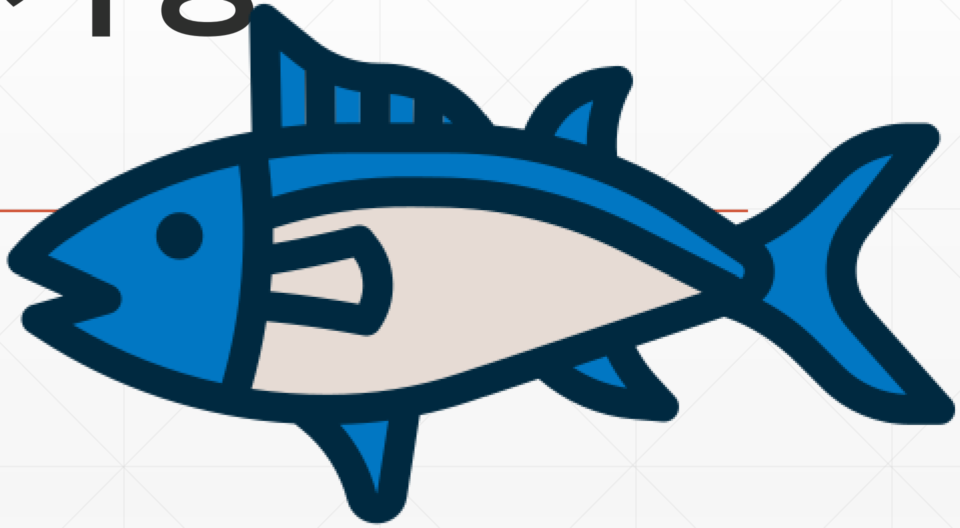
```python
for i in range(400,501):
  ans += math.comb(500,i)*bellNumber(i)
print(ans)
```

해당 문제의 모든 경우의 수의 길이는 846

1735800167186508423311299999469818039768868240921211702170101330325881232710676689
0503008112020253423939617254192428122132514221410771359030774885829040388117428413
2942663172294828573765019059627904499865668461699412160276607372350690304749027728
1055319120513428721772971516975927134371699702404925831135231241691603226613523417
6378910581754868532830484563364960747297925420924689685110439033727651247779775658
9881626836490961087947126464154969285484098267317272259387455474176961898212910098
9194320659761507813381576438575695284483576151494423865004488215082072526278245248
5572494930248112868815318654602137066532542450500699169265767746953564219666682805
4796944417365499048250420794051178172210163219845342512776193949122298967594994199
4731695363605420595109682290167689022875388234376198562810164113979959490584860387
61623536490422860173175019

결론

경우의 수는 구했는데...
종목별 펀드 구현은..?
지금의 사조 참치는 불가능



감사합니다.