딱 맞춰조

박스 추천 및 적재 최적화 알고리즘 구축

201520202 이민혁

201421589 배주영

201723224 윤형배

201520219 김현용

INDEX

주제 선정 배경

주제 선정 배경

택배회사의 과대 포장은 알바실수?...'택배시대' 대책이 필요한 택배회사들

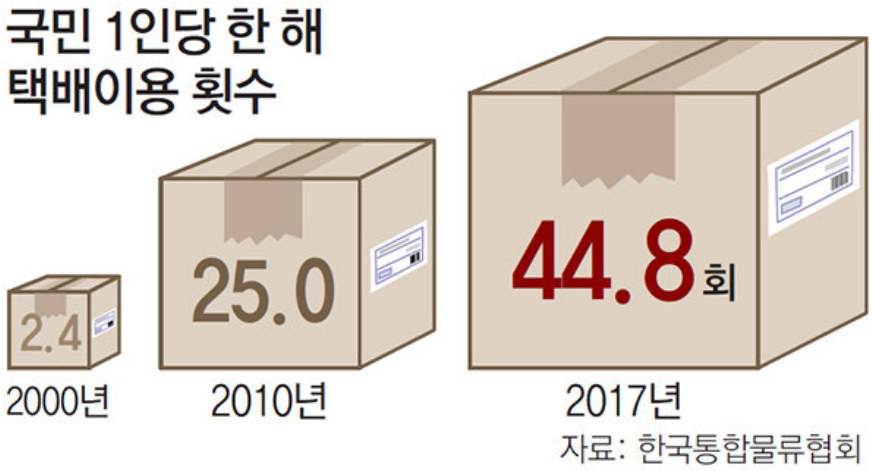
입력 2020.11.10 15:08 | 수정 2020.11.10 15:08

[한경잡앤조이=이도희 기자/이원지 대학생 기자] 물류 기술이 발전하면서 빠른 택배 서비스는 우리 생활에 빠질 수 없는 존재가 됐다. 특히 올해 코로나19 사태가 장기화됨에 따라 '언택트(Untact)' 문화가 일상이 되면서 온라인 쇼핑 이용량이 폭발적으로 증가하고 있다.

하지만 우리는 편리함의 대가로 환경에 큰 부담을 주고 있다. 각종 택배 포장으로 인한 생활 쓰레기가 급격하게 늘고 있는 것이 현실이다. 환경부에 따르면 올해 상반기 생활 폐기물은 4890톤에서 5349톤으로 늘어 지난해 같은 기간 대비 11.2% 증가했다. 특히 종이류는 687톤에서 889톤으로 증가해 총 23.9% 늘어난 것으로 조사됐다. 택배 물량 증가로 인해 종이 박스가 많이 사용된 것이 원인으로 파악됐다.

택배 물량이 늘어나기도 했지만 과대 포장도 심각한 문제다. 오래 전부터 이커머스 업계의 과대포장 문제는 꾸준히 제기돼 왔다. 제품 포장 과정에서 파손 방지를 위해 지나치게 많은 포장재를 낭비하고 있기 때문이다. OECD 공식 통계에 따르면 우리나라 포장폐기물 발생량은 미국 다음으로 타 국가들보다 월등히 많은 편이다. 이에 대한 감량화가 시급한 것으로 보인다.

친환경 시대, 유통업계에서는 플라스틱 팩 대신 종이팩으로 스티로폼 상자를 종이 상자로 대체하는 노력을 행하고 있다. 쿠팡은 신선식품 새벽 배송 서비스에 포장재 대신 보냉백을 이용한 '로켓프레시 에코' 서비스 운영을 시작했다. 일반 상품도 85%의 상품을 '박스리스'형태로 종이 골판지 상자 없이 포장하면서 폐기물을 줄이기 위한 대안을 내놓고 있다. 그러나 여전히 과대 포장 문제는 해결되지 않은 것으로 보인다.



주제 선정 배경

고객 주문 정보와 피킹된
상품 정보가 일치 하는지
작업자가 일일이 검수

- ✓ 오출 가능성 존재
- ✓ 비효율적 인력운용
- ✓ 낮은 생산성

주제 선정 배경

작업자의 판단에 따라
임의로 박스 크기 선택 후
상품을 적재

- ✓ 적재 시간 증가
- ✓ 박스 선택 오류 가능성
- ✓ 낮은 생산성

프로세스 분석

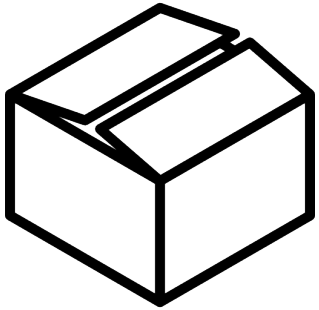
- 기존 프로세스
- 개선 프로세스

프로세스 분석 (기존 프로세스)

고객 주문

상품 피킹

작업자 상품 검수



박스 선택 및 적재

프로세스 분석 (개선 프로세스)

고객 주문

상품 피킹

Barcode 인식

최적화된 크기의 박스 추천

최적화된 상품 적재 방식 추천

프로세스 구축 및 구현

- 데이터 수집 및 DB구축
- Barcode 인식
- 박스 크기 추천
- 상품 적재 최적화

데이터 수집 및 DB 구축

	A	B	C	D	E	F	G
1	Number	Product	Width(cm)	Height(cm)	Depth(cm)	Weight(g)	Barcode Number
2	1	코멧 순백 3겹 라벤더 바닐라 롤 화장지	30.7	49	19.4	700	2491787579453
3	2	저자극 시그니처 아기 물티슈 엠보싱 캡형	20	15	8	350	5974512647044
4	3	EM 발효 숯 세안비누	7	6.5	2.4	100	1508875316943
5	4	1겹민자 롤형 핸드타올	20	20	5	60	2577525901777
6	5	샤오미 체지방 체중계 2세대	30	30	25	1700	1656386484740
7	6	깨끗한 2겹 무형광 핸드타올 (5000매)	50	42	30	1400	5521304664851
8	7	프라그란스비즈 차량용 방향제	15	20	7	180	7738923681618
9	8	네이쳐리빙 모던데일 3단 무빙 트롤리 수납선반	40	64	21.5	4500	8369893618065
10	9	모노블 더길어진 반전매력 재활용 분리수거함	30	60	30	800	4861308483906
11	10	키친 스테인레스 식기건조대 2단	42.5	38	26.5	2200	3907072375514
12	11	크린롤백 200매입 대	34	50	10	440	1648315648967
13	12	크린롤백 200매입 소	17	25	5	220	8659802298636
14	13	리벤스 코팅 논슬립 바지걸이	36	15	2	120	8018062662713
15	14	양키캔들 라지 자 미드썸머나잇향	10.5	17	8	623	8223949236875
16	15	디캣 차량용 방향제 클립형 30C 블랙체리 로즈골드	2.8	0.8	0.8	40	3831281131110
17	16	채움리빙 라탄 팬트리 수납 바구니 1호	44.6	31	18.5	120	5382954495315
18	17	기본에 크린장갑 500매	24	28	12	480	9698925351306
19	18	홈플래닛 접이식 다용도 쇼핑카트 블랙 L	42	40.5	8	2800	1741138965340
20	19	조은리빙 이동접이식 빨래건조대 대형	12	107	52	1800	2340798424957
21	20	샬블러썸 프레이그런스드 옷장방향제 퓨어블루밍츕	11	17	7	20	1674490895290
22	21	스카트 하루 한 장 The 버블 수세미 50p	20	22	20	330	1242981173847
23	22	탐사 베이킹소다 플러스 세탁세제	45	55	40	4000	1632819393645
24	23	리스테린 헬씨브라이트 구강세정제	15	35	6	750	3748272201979
25	24	코멧 음식물 쓰레기통 그레이	22	22	24	220	5497800454476
26	25	시그니처 디퓨너 250ml	7.5	7.5	20.7	550	5753325022907
27	26	스포츠 싱글 마사지볼	13.5	6.8	6.8	130	2372978690254
28	27	록키스 풋 케어 애기발 스틱 20g	8	12	4.2	160	2567382935225
29	28	이지롤 분리수거용 쓰레기봉투 20L	24	30	10	270	9212968914229
30	29	리템LnC 마이룸 우산꽂이 자석형	19	50	55	100	2443154503655
31	30	리버그린 흡착 심플선반	27	9	11	140	4477394156211
32	31	해피홈 리퀴드 훈증기 + 리필	38	24	12	620	6110412694342
33	32	다우니 엑스퍼트 실내 건조 1L	18	45	8	1800	4737517150825
34	33	봉쥬르 드링킹자 450ml 유리병 보관용기 컵	6	13	6	180	6916800623431
35	34	스마트보울 다용도 면기	16	9.5	16	150	7083224253835
36	35	심플쿡냉동밥전자렌지용기(400ml)	15.2	4	15.2	80	8032586313281
37	36	슬라이드 캡 쓰레기통 싱크대 걸이 휴지통	27	18	17	250	9501016910608
38	37	로지텍 블루투스 멀티 디바이스 키보드 K380	27.9	12.4	1.6	423	3938354211842

coupang

AUCTION.

Gmarket

Homeplus

데이터 수집 및 DB 구축

프로세스 구현 (Barcode 인식)

알고리즘

- 고객의 주문 내용 데이터 수신
- ⬇
- 데이터 베이스 바코드 정보 확인
- ⬇
- 제품 바코드 인식
- ⬇
- 주문 내용 데이터 일치 확인

프로세스 구현 (Barcode 인식)

고객의 주문 내용 데이터 수신

주문 내용 데이터 : 제품명과 주문 개수

	Product	Number
1	잘풀리는 집 보습 미용티수 230매	3
2	커피여과지 40개입	1
3	삼다수 2L	4

프로세스 구현 (Barcode 인식)

데이터 베이스 Barcode 정보 확인

제품 데이터 베이스

```
df = pd.read_excel('종설_제품데이터수집.xlsx')
df = pd.DataFrame(data=df)
df.head(10)
```

	Number	Product	Width(cm)	Height(cm)	Depth(cm)	Weight(g)	Barcode Number
0	1	코멧 순백 3겹 라벤더 바닐라 롤 화장지	30.7	49.0	19.4	700	2491787579453
1	2	저자극 시그니처 아기 물티슈 엠보싱 캡형	20.0	15.0	8.0	350	5974512647044
2	3	EM 발효 숯 세안비누	7.0	6.5	2.4	100	1508875316943
3	4	1겹민자 롤형 핸드타올	20.0	20.0	5.0	60	2577525901777
4	5	샤오미 체지방 체중계 2세대	30.0	30.0	25.0	1700	1656386484740
5	6	깨끗한 2겹 무형광 핸드타올 (5000매)	50.0	42.0	30.0	1400	5521304664851
6	7	프라그란스비즈 차량용 방향제	15.0	20.0	7.0	180	7738923681618
7	8	네이쳐리빙 모던데일 3단 무빙 트롤리 수납선반	40.0	64.0	21.5	4500	8369893618065
8	9	모노블 더길어진 반전매력 재활용 분리수거함	30.0	60.0	30.0	800	4861308483906
9	10	키친 스테인레스 식기건조대 2단	42.5	38.0	26.5	2200	3907072375514

주문 내용 데이터에 바코드 정보 추가

```
df2 = df.drop(['Number','Width(cm)', 'Height(cm)', 'Depth(cm)','Weight(g)'], axis=1)
df2.head(5)
```

	Product	Barcode Number
0	코멧 순백 3겹 라벤더 바닐라 롤 화장지	2491787579453
1	저자극 시그니처 아기 물티슈 엠보싱 캡형	5974512647044
2	EM 발효 숯 세안비누	1508875316943
3	1겹민자 롤형 핸드타올	2577525901777
4	샤오미 체지방 체중계 2세대	1656386484740

```
data = pd.merge(products, df2,how='left')
data = data.sort_values('Barcode Number')
data = data.reset_index()
data = data.drop(['index'], axis=1)
data
```

	Product	Number	Barcode Number
0	커피여과지 40개입	1	4006508122134
1	삼다수 2L	4	8808244201045
2	잘풀리는 집 보습 미용티슈 230매	3	8809180744146

프로세스 구현 (Barcode 인식)

제품 바코드 인식

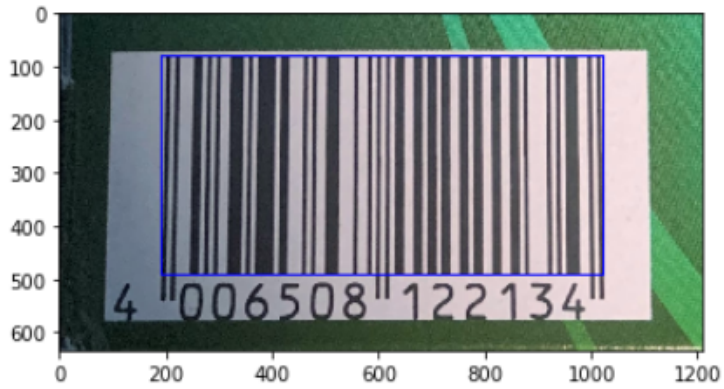
주문 제품의 개수 파악

```
Sum_Number = 0
for i in range(len(data)):
    Sum_Number += data.iloc[i,1]
Sum_Number
```

```
8
```

제품 바코드 인식

```
B_df = pd.DataFrame(columns=['Barcode'])

for i in range(Sum_Number):
    img_name = 'img%d.jpg'%i
    img = cv2.imread(img_name,cv2.COLOR_BGR2GRAY)
    decoded = pyzbar.decode(img)
    for d in decoded:
        #print(d.data.decode('utf-8'))
        cv2.rectangle(img, (d.rect[0], d.rect[1]),
                      (d.rect[0] + d.rect[2], d.rect[1] + d.rect[3]),
                      (0, 0, 255), 2)
        B_df.loc[i] = d.data.decode('utf-8')
        plt.imshow(img)
```



프로세스 구현 (Barcode 인식)

주문 내용 데이터 일치 확인

인식된 정보 데이터 프레임 구축

```
B_df_2 = pd.DataFrame(data=B_df['Barcode Number'].value_counts())
B_df_2 = B_df_2.reset_index()
B_df_2['Number'] = B_df_2['Barcode Number']
B_df_2['Barcode Number'] = B_df_2['index']
B_df_2 = B_df_2.sort_values('Barcode Number')
B_df_2 = B_df_2.reset_index()
B_df_2 = B_df_2.drop(['level_0'], axis=1)
B_df_2 = B_df_2.drop(['index'], axis=1)
B_df_2['Barcode Number'] = pd.to_numeric(B_df_2['Barcode Number'])
B_df_2
```

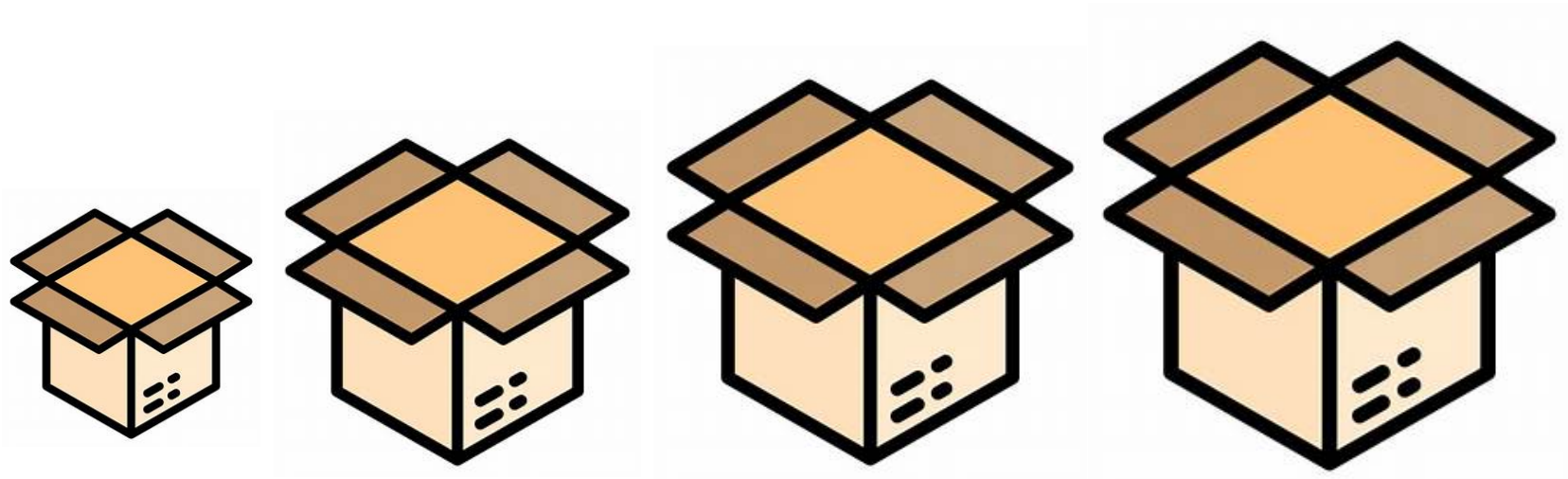
	Barcode Number	Number
0	4006508122134	1
1	8808244201045	4
2	8809180744146	3

주문 내용 데이터 일치 확인

```
data.iloc[:,1:3].eq(B_df_2)
```

	Barcode Number	Number
0	True	True
1	True	True
2	True	True

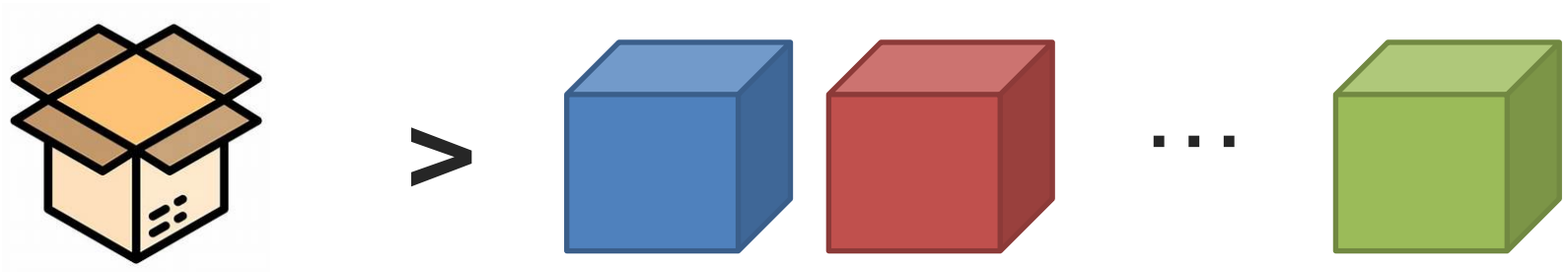
프로세스 구현 (박스 크기 추천)



상자 A　　상자 B　　상자 C　　상자 D

Box_List = [[22, 19, 9],[27,18,15],[32,25,10],[34,25,21],[41,31,28],[48,38,34]]

프로세스 구현 (박스 크기 추천)

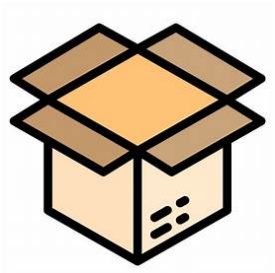


상자 A의 부피 > N개의 물품의 부피 합

True

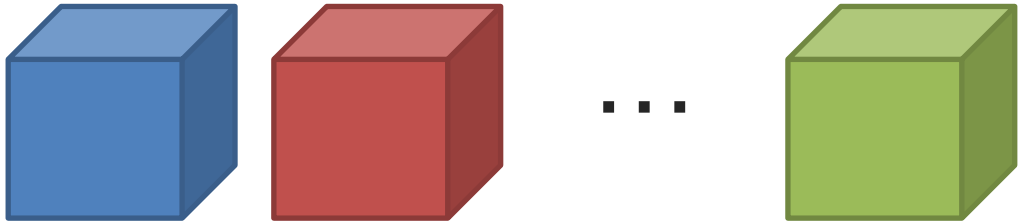
상자의 가로, 세로, 높이 > 물품들의 가로, 세로, 높이

프로세스 구현 (박스 크기 추천)



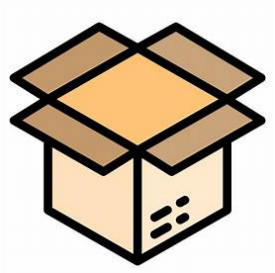
상자 A의 부피

>



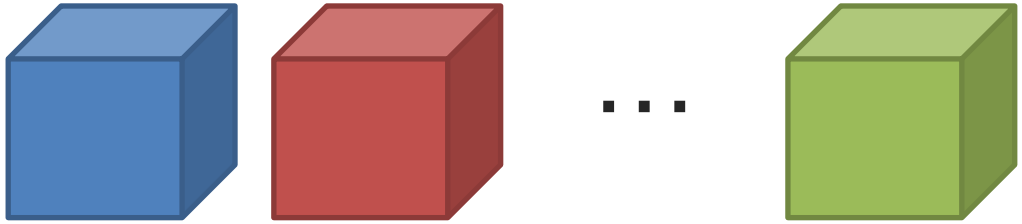
N개의 물품의 부피 합

Fals
e



상자 B의 부피

>



N개의 물품의 부피 합

프로세스 구현 (박스 크기 추천)

```
def box_select(box_list, product_list):
  for i in box_list:
    box_volum = i[0]*i[1]*i[2]
    box = i.copy()
    box.sort()
    product_volum_sum = 0
    for j in product_list:
      product_volum = j[0]*j[1]*j[2]
      product_volum_sum += product_volum_sum

    if box_volum < product_volum_sum:
      continue

    count = 0  #모든 상품의 모든 길이가 상자의 크기보다 작은지 확인 하기 위하여
    for k in product_list:
      k.sort()
      if k[0] <= box[0] and k[1] <= box[1] and k[2] <= box[2]:
        count += 1

    if len(product_list) == count:
      selected = i
      break

  return selected
```

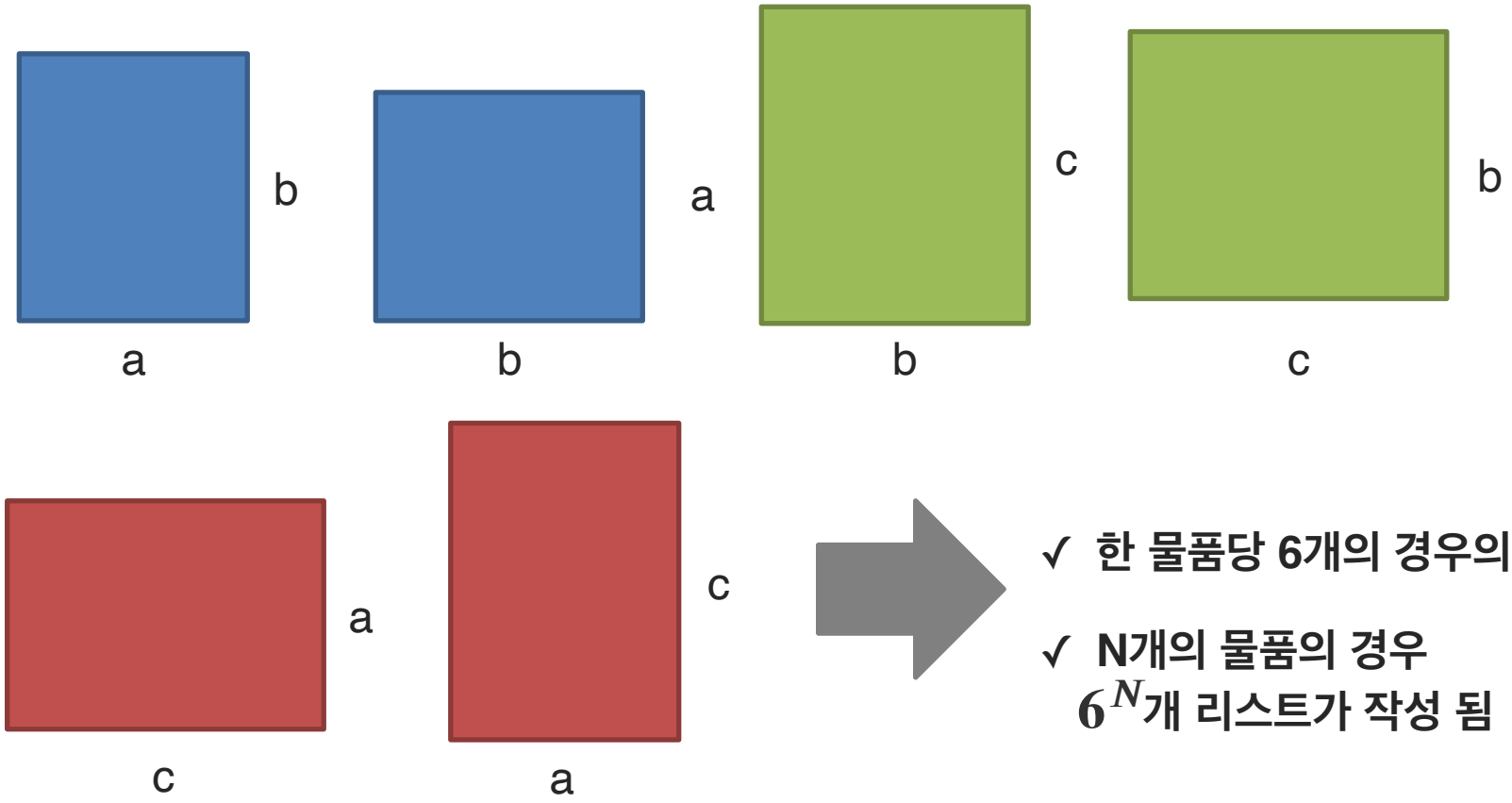
상자의 부피와
물품의 부피 합 비교

상자의 가로,세로,높이와
물품의 가로,세로,높이 비교

선택된 상자 정보값 리턴

프로세스 구현 (상품 적재 최적화)

물건 회전 리스트



✓ 한 물품당 6개의 경우의 수

✓ N개의 물품의 경우
6^N개 리스트가 작성 됨

프로세스 구현 (상품 적재 최적화)

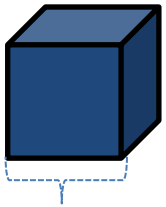
```
def make_list(product_list):
  list = []
  for i in product_list:
    case = [[i[0],i[1],i[2]],[i[0],i[2],i[1]],[i[1],i[0],i[2]],
            [i[1],i[2],i[0]],[i[2],i[0],i[1]],[i[2],i[1],i[0]]]
    list.append(case)
  #각 상품 마다 모든 회전된 리스트를 먼저 만들었음

  n=len(list)
  list_case=[]
  a = 6**n
  #6의 n승번 반복하도록 하는 거
  for i in range(a):
    list_add=[]
    for i in range(n):
      y = random.randint(0,5)
      a = list[i][y]
      list_add.append(a)
  #회전된 리스트를 가지고 6^n번 리스트를 만들었다.

    list_case.append(list_add)

  return list_case
```

각 상품 별로 회전된 리스트를 만든 과정

회전 리스트를 통해 6^N 개의 경우의 수를 가진 리스트 작성

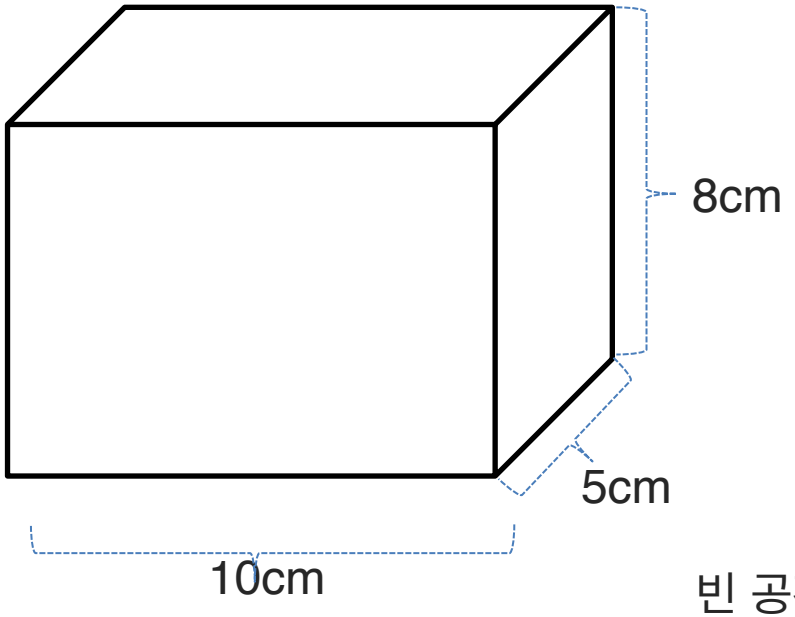
경우의 수 리스트 리턴

프로세스 구현 (상품 적재 최적화)



1cm

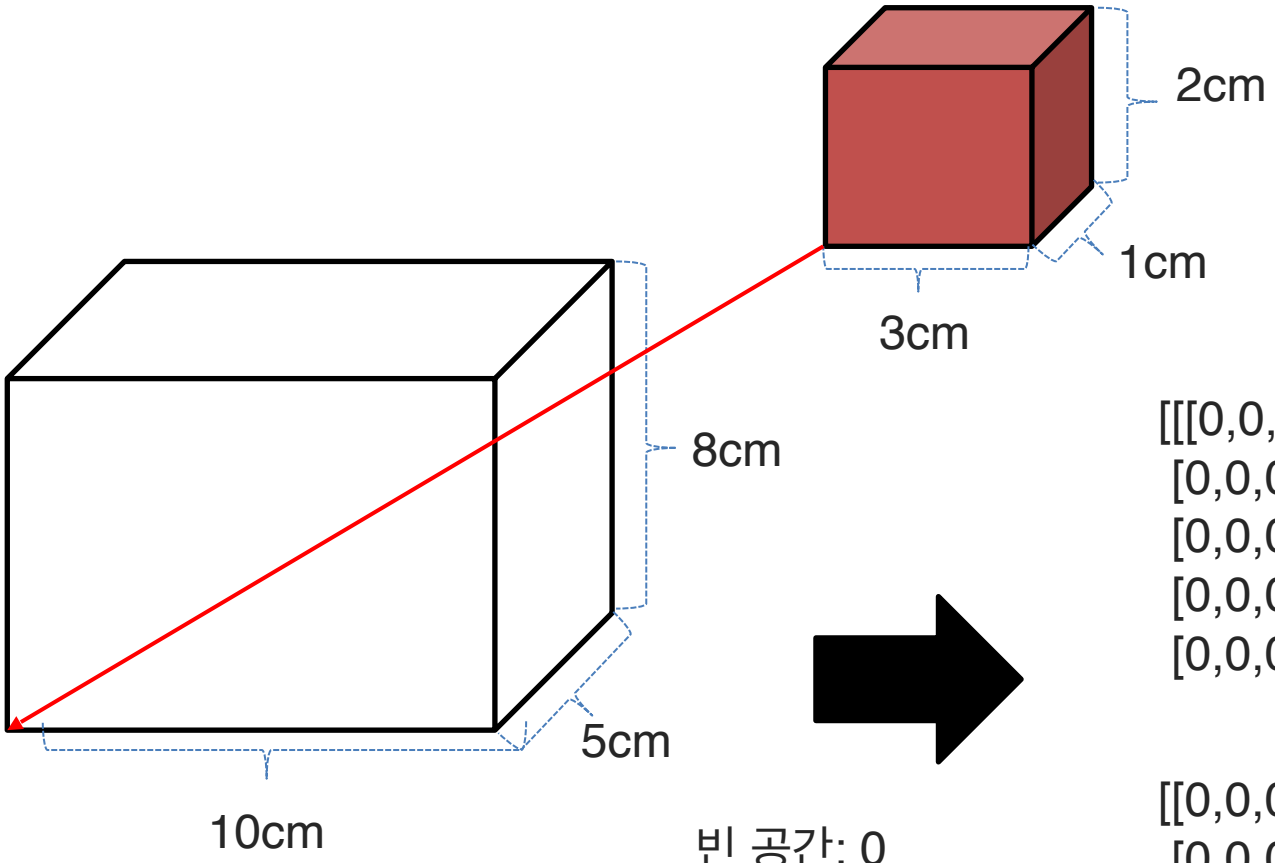
$1cm^3$인 정육면체가
10 x 5 x 8개가 나오게 됨



빈 공간: 0
채워진 공간: 1

```
[[[0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0]]
        .
        .
[[0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0]]]
```

프로세스 구현 (상품 적재 최적화)



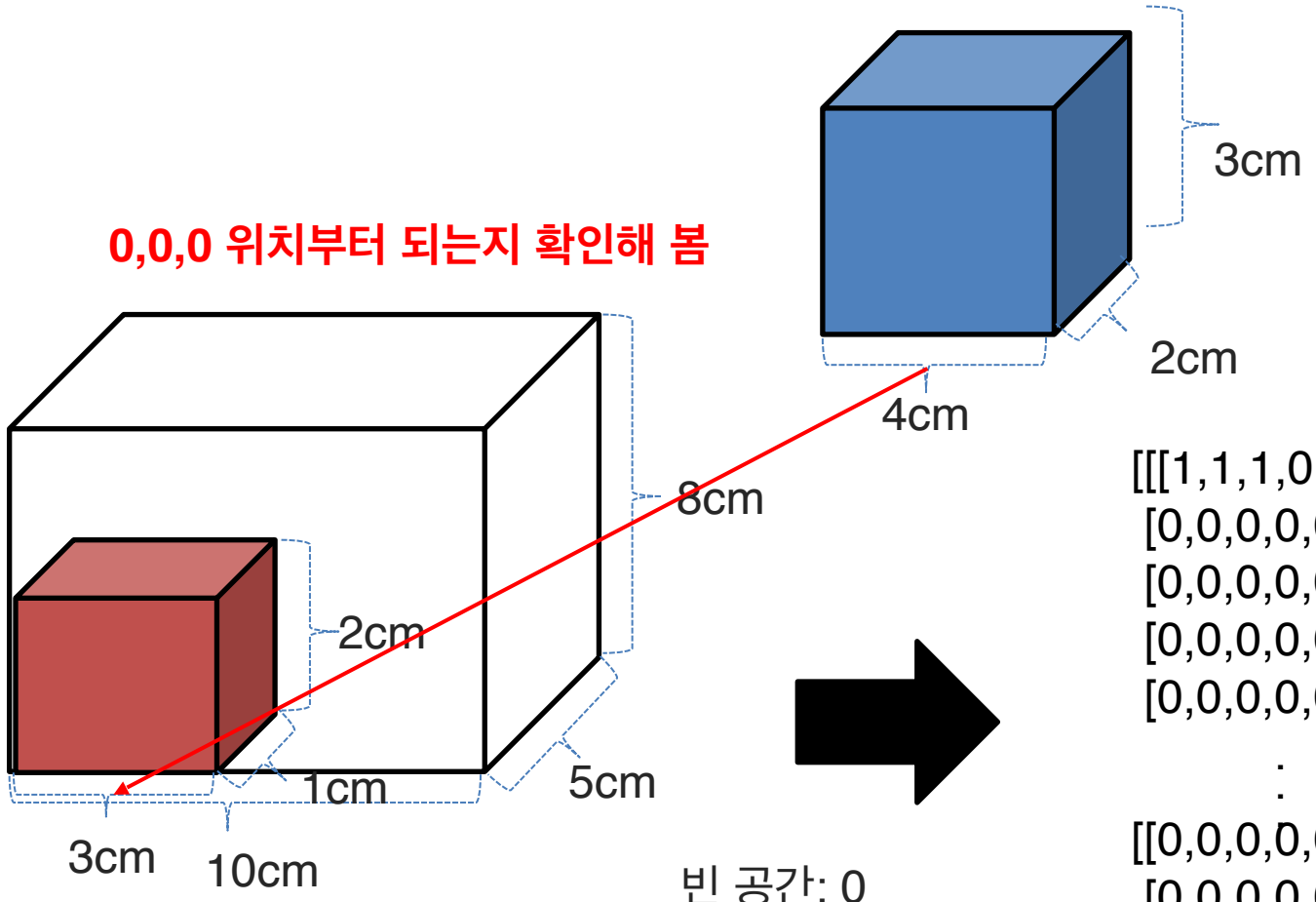
빈 공간: 0
채워진 공간: 1

[[[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0]]

.

[[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0]]]

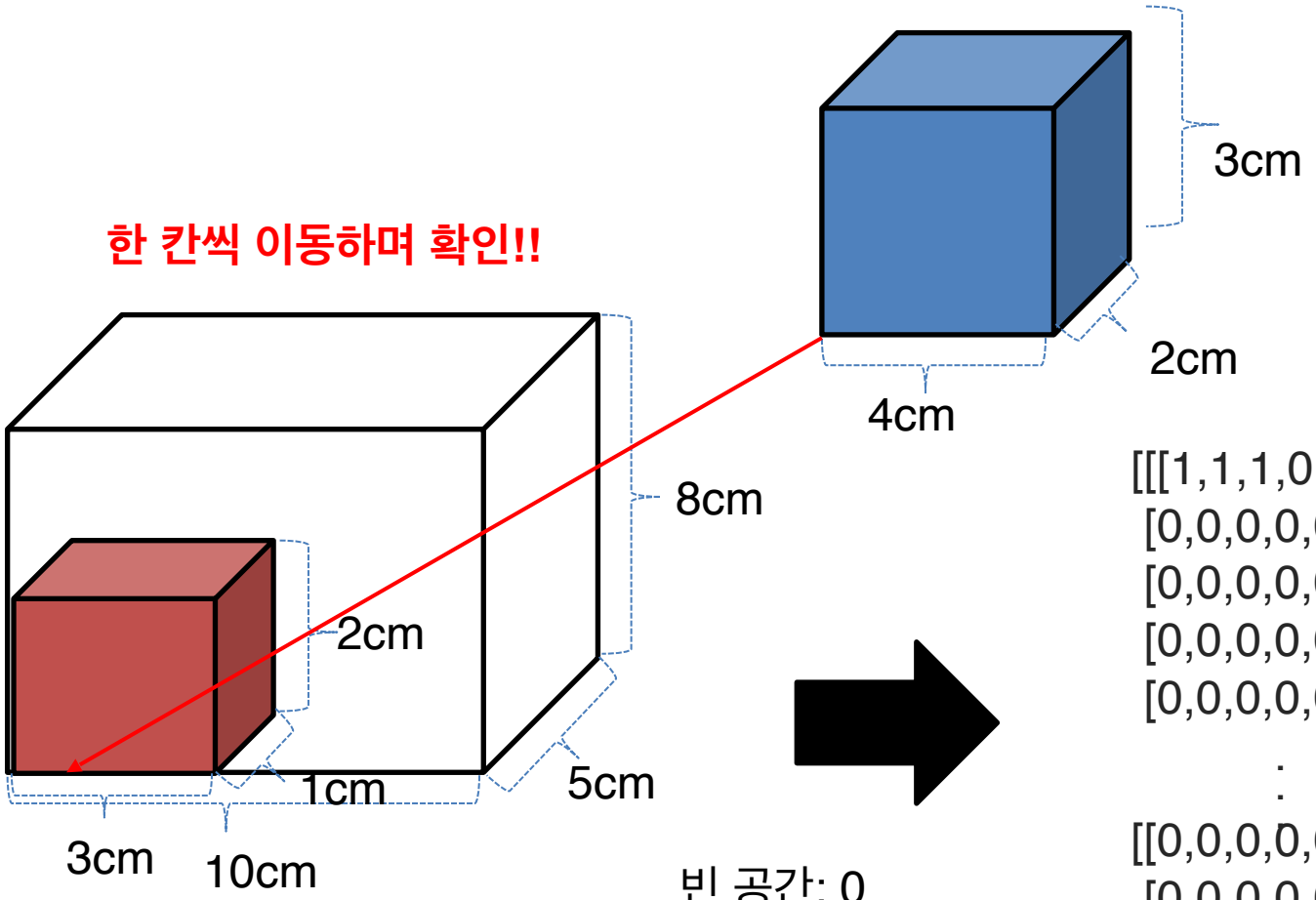
프로세스 구현 (상품 적재 최적화)



빈 공간: 0
채워진 공간: 1

```
[[[1,1,1,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0]]
        .
        .
[[0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0]]]
```

프로세스 구현 (상품 적재 최적화)

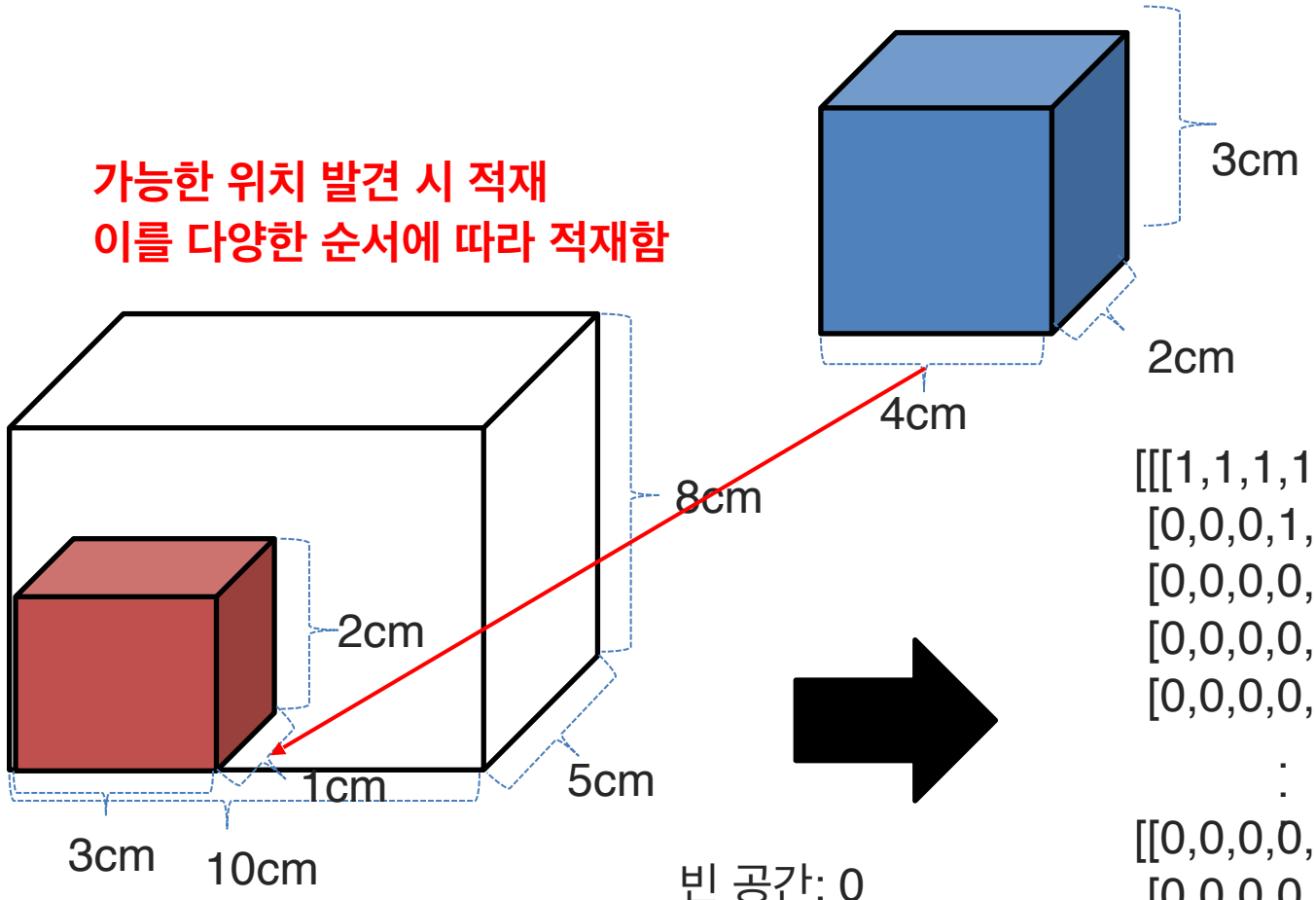


빈 공간: 0
채워진 공간: 1

```
[[[1,1,1,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0]]
          .
          .
[[0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0]]]
```

프로세스 구현 (상품 적재 최적화)

가능한 위치 발견 시 적재
이를 다양한 순서에 따라 적재함



빈 공간: 0
채워진 공간: 1

```
[[[1,1,1,1,1,1,1,0,0,0],
 [0,0,0,1,1,1,1,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0]]
          :
[[0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0]]]
```

프로세스 구현 (상품 적재 최적화)

```
def load(box,list_case):
    box_size = np.zeros((box[2],box[1],box[0]))#높이, 세로, 가로
    n = len(list_case[0]) # 하나의 케이스 길이
    product_information = []
    print(n)
    for i in list_case:
        *b, = itertools.permutations(i,n) #각 케이스의 조합을 만듬
        for j in b: #조합의 첫번째 부터 마지막 까지 각각의 경우에 대해서 생각해보는(아마 n!개가 나올 거임)
            position = np.array([0,0,0])  #[x,y,z]
            can = True
            pos_hist=[]
            box_size = np.zeros((box[2],box[1],box[0]))
            for case in j: #각 조합의 첫번째 부터 적재가 가능한지 검사를 해 본다.
                position = np.array([0,0,0])

                while(True):

                #실질적으로 적재가 가능한지 검사하는 코드
                    slice_value = box_size[position[2]:case[2]+position[2],
                                           position[1]:case[1]+position[1],position[0]:case[0]+position[0]]
                    correct = np.zeros((case[2],case[1],case[0]))

                    #자른 만큼의 크기와 동일하게 0으로 되어있다면 1로 채워 넣음
                    if np.array_equal(slice_value,correct):
                        box_size[position[2]:case[2]+position[2],position[1]:case[1]+position[1],
                                 position[0]:case[0]+position[0]] = np.ones((case[2],case[1],case[0]))
                        pos_hist.append(position)

                        break
```

구현하는데 필요한 기본값 설정

적재가능 여부 검사

프로세스 구현 (상품 적재 최적화)

```
    #적재가 불가능 할 경우 한칸씩 이동해 보며 확인을 해보는 경우를 생각해 보려고 함
     if position[0] < box[0]-case[0]:
       position[0] += 1

     elif position[1] < box[1]-case[1]:
       position[1] += 1
       if position[0] == box[0]-case[0]:
         position[0] = 0

     elif position[2] < box[2] - case[2]:
       position[2] += 1
       if position[0] == box[0]-case[0]:
         position[0] = 0

       if position[1] == box[1]-case[1]:
         position[1] = 0
```

적재가 불가능할 경우 위치를 이동해 보며 확인하는 부분

```
     if position[0] >= box[0] - case[0] and position[1] >= box[1] - case[1]
       and position[2] >= box[2] - case[2]:
       can = False
       break
```

적재 불가능 여부 판단

```
    if can == False:
      break


    if len(pos_hist) == n:
      break

  if len(pos_hist) == n:
     product_information = j
     break
if len(pos_hist) == n:
  break

return pos_hist,product_information
```

물품의 위치와 가로, 세로, 높이 정보 리턴

프로세스 구현 (상품 적재 최적화)

```
from vpython import*
```

Vpython을 이용하여 시각화 하였음

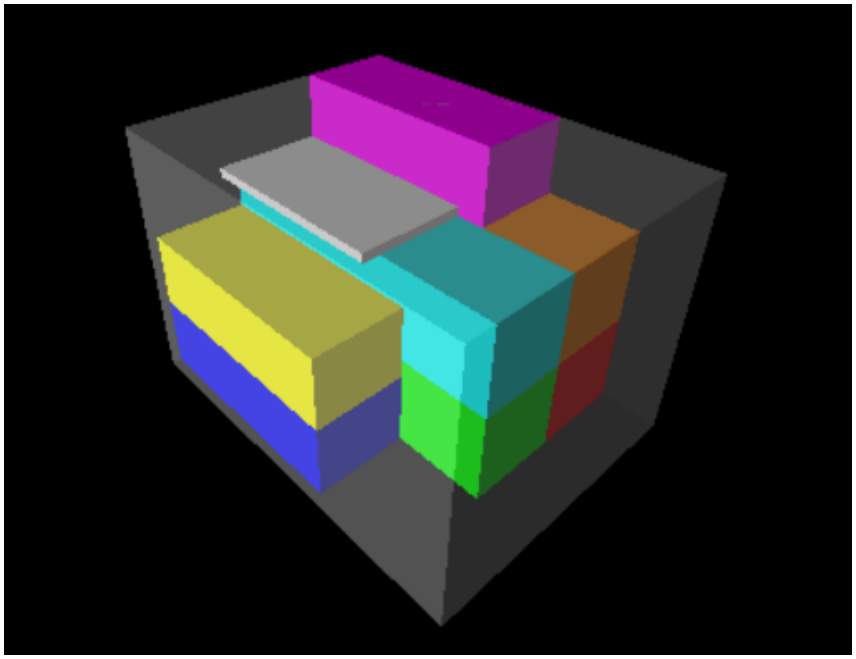
```
center = vector(box_size[0]/2,box_size[2]/2,box_size[1]/2)
size = vector(box_size[0],box_size[2],box_size[1])
box_big = box(pos = center, size = size, color = color.white, opacity = 0.3)
```

상자 시각화

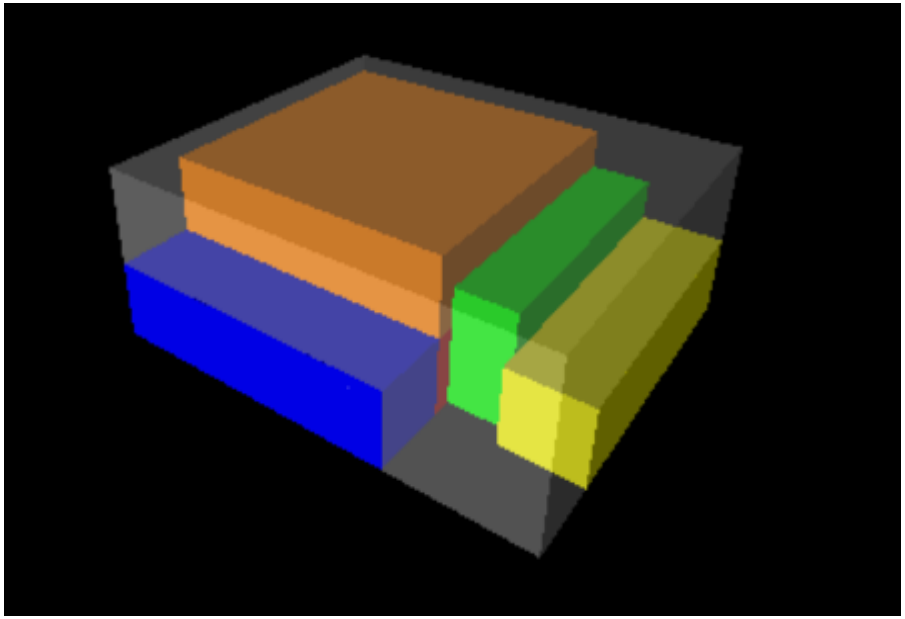
```
print(hist[0])
print(product_inform[0])
center = vector(hist[0][0]+product_inform[0][0]/2,hist[0][2]+product_inform[0][2]/2,hist[0][1]+product_inform[0][1]/2)
size = vector(product_inform[0][0],product_inform[0][2],product_inform[0][1])
box_big = box(pos = center, size = size, color = color.blue )

color_list = [(1,0,0),(0,1,0),(0,0,1),(1,1,0),(1,0.5,0),(0,1,1),(1,0,1),(1,1,1)]

for i in range(len(hist)):
    s = i%8
    center = vector(hist[i][0]+product_inform[i][0]/2,hist[i][2]+product_inform[i][2]/2,hist[i][1]+product_inform[i][1]/2)
    size = vector(product_inform[i][0],product_inform[i][2],product_inform[i][1])
    box_big = box(pos = center, size = size, color = vector(color_list[s][0],color_list[s][1],color_list[s][2]))
```

물품의 적재 상황 시각화

프로세스 구현 (상품 적재 최적화)



Box_size = [41, 31, 28]
product_list = [[35,10,10],[35,10,10],[35,10,10],[35,10,10]
[25,10,8],[25,10,8],[25,10,8],[22,12,1]]



Box_size = [22, 19, 9]
product_list = [[15,4,15],[6,14,3],[15,4,4],[15,4,4],[15,4,15]]

의의 및 결론

의의 및 결론

1 기술의 범용화 및 시스템화

- ✓ 물류 자동화의 단계가 매우 높은 일부 대기업의 유통 물류센터를 제외하고는 비용 및 시간의 이슈로 인해 아직까지 물류 자동화 시스템에 대한 구축 미흡
- ✓ 박스 선정, 상품 적재 및 검수의 과정이 작업자의 직관에 따라 결정되는 경우가 많은데, 본 시스템을 통해 상품 데이터 및 고객 주문정보만 있으면 누구나 쉽게 자동화 프로세스를 구축하고 빠르게 도입할 수 있음.

2 상품 적재 최적화를 통한 물류 효율화

- ✓ 코로나로 인해 E-COMMERCE의 급속한 성장으로 물류 자동화에 대한 중요도가 높아짐. 고객 주문 정보의 유입부터 최종 출하의 과정까지 일부 자동화 시스템을 구축함으로써 상품 적재를 최적화하고 물류 효율성 확보
- ✓ 상품 데이터베이스를 통해 시스템이 사전에 작업자에게 최적의 박스 크기 및 적재 방식을 추천해줌으로써 물류에서의 공간 효율 및 인력 운영 효율 증진

의의 및 결론

3 오출 방지를 통한 고객 서비스 개선

- ✓ 대부분의 유통 물류센터의 경우 작업자가 출고지시서와 실제 picking된 물품을 직접 육안으로 확인하며 검수의 과정을 거침. 숙련된 작업자임에도 불구하고 편의점 유통물류센터 기준 약 5-7% 사이의 오출 존재

- ✓ 카메라 센서를 통해 상품 바코드를 인식하고 이를 실제 고객 주문 정보와 대조함으로써 고객이 주문한 상품이 맞는지에 대한 검수의 과정을 시스템이 대체. 상품 출고 정확도 100% 달성 가능.

4 자원 소모 최소화 및 ESG 확보

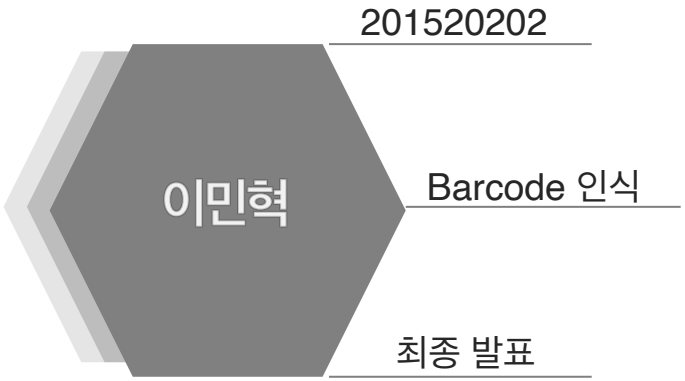
- ✓ 공정 및 물류단계에서의 불필요한 자원 소모 최소화를 통해 상품의 원가경쟁력 확보 및 수익성 개선

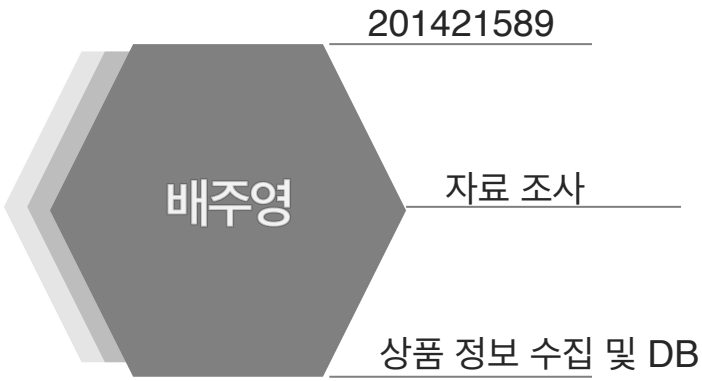
- ✓ ESG(Environmental, Social and Governance)가 글로벌 트렌드로 자리매김하면서 친환경 경영의 중요도가 높아짐. 최적의 자원 사용으로 녹색 물류를 실천하고 친환경 및 ESG 관련 긍정적 요소 확보

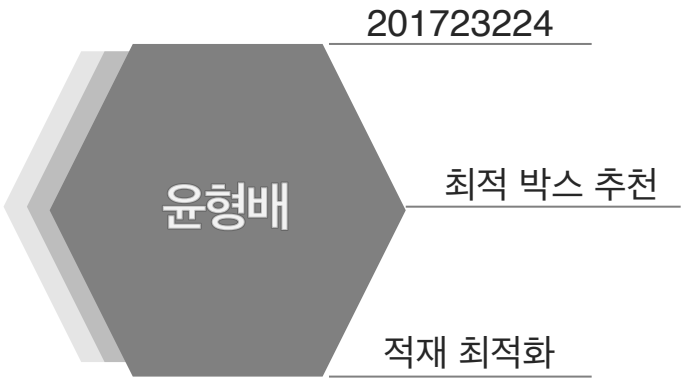
기타

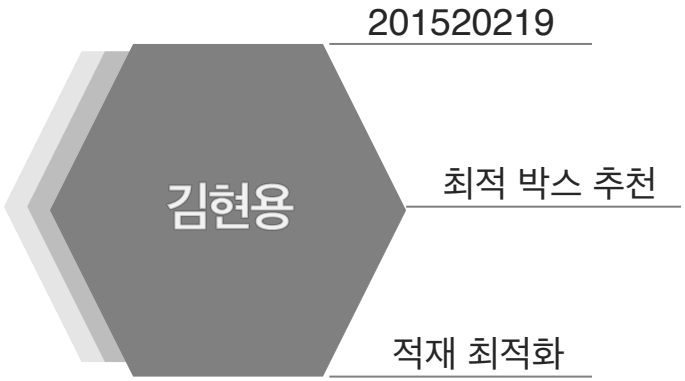
- 역할 분담
- 주차 별 진행 내용

역할 분담


201520202
이민혁
Barcode 인식
최종 발표


201421589
배주영
자료 조사
상품 정보 수집 및 DB 구축


201723224
윤형배
최적 박스 추천
적재 최적화


201520219
김현용
최적 박스 추천
적재 최적화

주차 별 진행 내용

주	내용
1주차	주제 선정 및 필요 기술 탐색
2주차	현황 파악 및 프로세스 분석
3주차	상품 및 박스, 고객 정보 관련 데이터베이스 구축
4주차	상품 Barcode 인식 관련 알고리즘 구축
5주차	고객 주문 별 최적의 박스 추천 알고리즘 구축
6주차	상품 적재 최적화 알고리즘 구축
7주차	전체 시스템 연동성 확보, 오류 도출 및 개선
8주차	최종 결과물 구현 및 검증
9주차	PPT 제작 및 최종 발표 준비

감사합니다