以下为第一问的python代码部分：

import pandas as pd

import numpy as np

df1=pd.read\_excel('附件1 近5年402家供应商的相关数据.xlsx',sheet\_name='企业的订货量（m³）')#可以通过sheet\_name来指定读取的表单

data1=df1.head()#默认读取前5行的数据

print("获取到所有的值:\n{0}".format(data1))#格式化输出

df2=pd.read\_excel('附件1 近5年402家供应商的相关数据.xlsx',sheet\_name='供应商的供货量（m³）')#可以通过sheet\_name来指定读取的表单

data2=df2.head()#默认读取前5行的数据

print("获取到所有的值:\n{0}".format(data2))#格式化输出

#求相关系数

df2\_new\_abc=df2.copy()#记录材料

df1.drop('供应商ID',axis=1,inplace=True)

df1.drop('材料分类',axis=1,inplace=True)

df2.drop('供应商ID',axis=1,inplace=True)

df2.drop('材料分类',axis=1,inplace=True)

print(df1)

print(df2)

corr=[]#相关系数

for i in range(0,402):

corr.append(df1.loc[i].corr(df2.loc[i],method='kendall'))

print(corr)

from numpy import \*

df2\_new=df2.copy()

df2\_new['变异系数']=df2.apply(lambda x:x.std(),axis =1)/df2.apply(lambda x:x.mean(),axis =1)

df2\_new['变异系数']=df2\_new['变异系数']/sqrt(240)

# annualVolatility = std(logreturns) / mean(logreturns)

# annualVolatility = annualVolatility / sqrt(1/252)

df2\_new["总和"] =df2.apply(lambda x:x.sum(),axis =1)

df2\_new["均值"] =df2.apply(lambda x:x.mean(),axis =1)

df2\_new["标准差"] =df2.apply(lambda x:x.mean(),axis =1)

df2\_new['零值率']=(df2 == 0).sum(axis=1)

df2\_new

df2\_new02=df2.T

df2\_new02

def regularit(df2\_new02):

df2\_new03 = pd.DataFrame(index=df2\_new02.index)

columns = df2\_new02.columns.tolist()

for c in columns:

d = df2\_new02[c]

MAX = d.max()

MIN = d.min()

df2\_new03[c] = ((d - MIN) / (MAX - MIN)).tolist()

return df2\_new03

df2\_new03=regularit(df2\_new02)

#df2\_new03=df2\_new03.T

df2\_new03

df2\_new['波动率']=''

df2\_new

np.quantile(df2\_new03[1], 0.9,axis=0)

df2\_new.loc[1,'波动率'] = 1

#df2\_new['波动率'][1]='1'

df2\_new

#print((np.quantile(df2\_new03[1], 0.9,axis=0) - np.quantile(df2\_new03[1], 0.1, axis=0) + 1) / (np.quantile(df2\_new03[1],0.1, axis=0) + 1))

for i in range(0,402):

df2\_new.loc[i,'波动率'] =np.quantile(df2\_new03[i], 0.9,axis=0) - np.quantile(df2\_new03[i], 0.1, axis=0)

#df2\_new['波动率'].apply(lambda x:(np.quantile(df2\_new03[x], 0.9,axis=0) - np.quantile(df2\_new03[x], 0.1, axis=0) + 1) / (np.quantile(df2\_new03[x],0.1, axis=0) + 1)

# df2\_new['波动率'][1]=(np.quantile(df2\_new03[1], 0.9,axis=0) - np.quantile(df2\_new03[1], 0.1, axis=0) + 1) / (df2\_new03[1],0.1, axis=0) + 1)

df2\_new

# df.loc[(df.traj\_id == id), 'quantile\_lng'] = (np.quantile(df.loc[(df.traj\_id == id), 'norm\_lng'], 0.9,

# axis=0) - np.quantile(

# df.loc[(df.traj\_id == id), 'norm\_lng'], 0.1, axis=0) + 1) / (

# np.quantile(df.loc[(df.traj\_id == id), 'norm\_lng'],

# 0.1, axis=0) + 1)

order\_data = pd.read\_excel('附件1 近5年402家供应商的相关数据.xlsx',sheet\_name = 0)

provide\_data = pd.read\_excel('附件1 近5年402家供应商的相关数据.xlsx',sheet\_name=1)

#print(order\_data)

#print(provide\_data)

turn = {'A':0.6,'B':0.66,'C':0.72}

t = provide\_data.set\_index(['供应商ID','材料分类']).T

sum\_new=[]#换算产量后的总和

for i in t.columns[0:]:

sum\_new.append(round(((t[i].sum()/turn[i[1]])), 2))

#print(i,t[i].sum()/turn[i[1]])

sum\_new#一个list,存储换算后总量

df2\_new['总和\_new']=sum\_new

df2\_new

df2\_new['均值\_new']=df2\_new['总和\_new']/(240-df2\_new['零值率'])

df2\_new['相关系数']=corr

df2\_new

df\_final=df2\_new[['相关系数','变异系数','波动率','总和\_new','均值\_new','零值率']]

df\_final

df\_zero=pd.read\_excel('连续为0的数量.xlsx')#可以通过sheet\_name来指定读取的表单

list\_0=[]

list\_0=df\_zero['连续为0量'].tolist()

list\_0

df\_final=df\_final.copy()

df\_final['连0数']=list\_0

df\_final=df\_final.T

df\_final.T

df\_final1=df\_final.T

df\_final1["变异系数"]=1-df\_final1["变异系数"]

df\_final1["波动率"]=1-df\_final1["波动率"]

df\_final1["总和\_new"]=(df\_final1["总和\_new"]-38.89)/(591438.3)

df\_final1["均值\_new"]=df\_final1["均值\_new"]/28200

df\_final1["零值率"]=1-df\_final1["零值率"]/240

df\_final1["连0数"]=e\*\*(-0.1\*(df\_final1["连0数"]-5))

df\_final1.reset\_index(drop=True, inplace=True)

A1=[0.18003228, 0.06824309\*0.5, 0.06824309\*0.5, 0.64105209\*0.75, 0.64105209\*0.25, 0.06824309\*0.67, 0.11067253\*0.33]

# [0.18003228 0.11067253 0.64105209 0.06824309]

# A1=[0.23230307,0.068862305,0.0688623505,0.4096642725,0.1365547575,0.05583552667,0.0279177633]

B = np.zeros((1, df\_final1.shape[1]))

df\_final1.shape[0]

df\_final2=df\_final1.T

df\_final2.shape[1]

df\_final2.shape[0]

df\_final2

row=3

column=5

min(A1[row-1], df\_final2[row][column])

list=[]

list.append(min(A1[row-1], df\_final2[row][column]))

list

def mymin(list):

for index in range(1, len(list)):

if index == 1:

temp = min(1, list[0]+list[1])

else:

temp = min(1, temp+list[index])

return temp

list\_final=[]

for column in range(0,df\_final2.shape[1]):#column 402 row 7

list = []

for row in range(0, df\_final2.shape[0]):

list.append(A1[row]\*df\_final2[column][row])

list\_final.append(mymin(list))

list\_final

df\_score1=df\_score1.sort\_values(by=0, ascending=False)

#df\_score1.to\_excel("score1.xlsx")

df\_score1=df\_score1.reset\_index(drop=True)

df\_score1

df2\_new['score']=list\_final

df2\_new['材料分类']=df2\_new\_abc['材料分类']

df2\_new['供应商ID']=df2\_new\_abc['供应商ID']

df2\_new=df2\_new.sort\_values(by='score', ascending=False)

df2\_new

df2\_new['供应商ID'].to\_excel("df\_score\_finnal.xlsx",index\_label=True)

以下为第二问的C++代码部分：

#include<iostream>

#include<vector>

#include<queue>

#include<map>

using namespace std;

class Solution {

private:

vector<double> d;//表示1m^3货物的制造产能

vector<double> e;//表示订购单位j种货物的价格

vector<double> m;//表示k周总供货量

vector<double> z;//表示k周总损耗

vector<double> w;//表示k周总成本

vector<double> u\_1;//表示k周采购成本

vector<double> u\_2;//表示k周运输成本

vector<double> u\_3;//表示k周存储成本

double f0, e0, g0;//运输、存储、损耗

//仓库货物时空分布图

int weeks;

double capacity\_need;

vector<vector<int>> a;//表示在第k周第i家供货商的供货量

vector<vector<int>> a\_;

vector<int> kind\_dict;//供货商商品种类映射 A-1,B-2,C-3

vector<double> actual\_consume;//A-0.6,B-0.66,C-0.72

vector<vector<vector<priority\_queue<pair<pair<int, int>,int>, vector<pair<pair<int, int>,int>>>>>> allocate\_status\_map;//[周][仓库]->(（周数，商家）,数量)

vector<vector<vector<int>>> block;//[周][仓库号]仓储分区(A-B-C-EMPTY)

vector<int> weekly\_actual\_capacity;//实际产能--按周记录

vector<vector<double>> b;//表示在k周第L家转运商的转运损耗率

vector<double> cumsum;//周累积和边界检测，需要在保证每周产能的前提下减少购买成本和贮存成本

vector<bool> useful\_bound;

vector<vector<int>> allocat\_ans;

double ave;

public:

Solution() {

ave = 0;

weeks = 24;

capacity\_need = 28200.\*2;

//kind\_dict = { 3,3,3,3,2,3,3,2,1,1,1,2,3,1,2,2,1,2,3,1,1,1,1,3,1,1,3,2,3,3,1,3,1,1,3,1,2,1,3,1,2,2,2,3,2,3,1,1,1,2 };//2

kind\_dict = { 3,3,3,3,2,3,3,2,1,1,1,2,3,1,2,2,1,2,3,1,1,1,1,3,1,1,3,2,3,3,1,3,1,1,3,1,2,1,3,1,2,2,2,3,2,3,1,1,1,2,

1,1,2,1,1,1,1,2,2,1,1,3,1,1,2,3,2,2,2,1,3,3,1,1,2,1,3,1,3,2,1,1,3,3,2,2,1,1,3,2,3,3,3,1,1,1,3,3,2,3};

actual\_consume = { 0.6,0.66,0.72 };

block = vector<vector<vector<int>>>(weeks, vector<vector<int>>(50, vector<int>(4, 0)));

for (int i = 0; i < block.size(); i++) {

for (int j = 0; j < block[i].size(); j++) {

block[i][j][3] = 6000;

}

}

weekly\_actual\_capacity = vector<int>(weeks, 0);

allocate\_status\_map = vector<vector<vector<priority\_queue<pair<pair<int, int>, int>, vector<pair<pair<int, int>, int>>>>>>(weeks, vector<vector<priority\_queue<pair<pair<int, int>, int>, vector<pair<pair<int, int>, int>>>>>(8, vector<priority\_queue<pair<pair<int, int>, int>, vector<pair<pair<int, int>, int>>>>(3)));

//a = { {1475,1276,10,6245,265,533,666,677,748,426,53,386,662,581,567,373,491,368,206,62,158,226,48,1185,139,352,109,12,81,37,63,0,36,1008,30,3,26,64,103,7,62,1,4,15,11,38,44,0,0,0},{1438,1006,12,833,219,569,779,808,853,261,6931,526,674,483,613,428,544,423,401,65,196,246,50,747,175,293,125,12,110,155,189,0,115,53,26,7,149,74,102,1221,19,5,1,23,11,70,74,2,2,1},{1376,1621,11,749,201,527,753,766,995,205,78,373,679,426,637,385,548,432,273,71,193,273,476,518,201,309,347,1682,486,37,51,0,45,56,19,296,32,83,148,7,27,71,5,12,12,35,127,1,0,1},{1459,1501,13421,862,227,618,861,817,1075,313,82,472,810,420,646,457,606,476,477,77,159,282,279,402,213,198,112,12,229,186,145,0,141,60,20,8,168,87,112,7,31,417,2,1,13,36,146,0,2,1},{1612,936,24,889,329,561,877,909,1118,6951,98,445,922,387,666,400,677,510,115,82,188,323,8781,344,217,178,143,24,104,56,52,0,46,69,28,11,36,100,111,19,60,10,7,1,14,32,739,0,0,2},{1573,1148,17,813,468,565,736,729,927,486,96,414,761,392,527,455,562,409,560,91,259,268,85,320,184,145,115,14,103,150,124,0,112,58,73,9,139,84,95,7,348,9,6,14,16,43,392,1,1,1},{1874,1144,18,740,605,511,765,789,890,445,94,491,585,691,525,420,521,408,120,244,573,256,59,317,166,143,115,20,83,48,44,0,38,59,55,11,29,67,95,14,1652,8,4,5,18,25,137,5,0,1},{2068,1224,24,854,697,609,801,834,1004,307,104,507,786,708,554,660,571,436,408,463,280,268,49,315,192,157,120,22,83,141,154,0,105,47,76,18,135,78,101,15,1690,16,3,6,21,44,25,1,0,2},{1951,1454,14,747,757,513,706,770,963,426,87,385,697,727,555,664,578,451,201,1839,200,276,48,316,185,158,119,17,98,52,50,359,40,51,79,8,29,76,97,11,1472,5,4,5,24,34,110,2,1,2},{1923,1261,17,818,727,628,691,695,876,314,92,337,690,842,499,672,544,391,413,481,147,242,46,298,186,135,103,14,86,127,177,587,94,45,68,9,116,77,85,11,1196,8,5,4,28,36,83,2,0,1},{2098,1519,17,727,787,527,751,699,815,372,86,472,738,804,497,562,538,408,168,396,128,254,46,288,164,150,118,15,90,44,65,431,37,47,77,8,29,79,89,12,1969,6,59,5,32,23,35,0,0,1},{2783,991,17,825,937,679,724,789,898,294,94,286,720,1111,521,589,552,395,457,202,145,255,45,288,171,176,121,17,80,125,155,73,89,52,66,11,117,85,89,11,2036,10,5,4,35,37,34,1,1,2},{1824,1252,19,798,949,571,819,804,989,306,89,385,777,969,557,563,609,436,107,264,95,277,45,299,182,159,112,17,98,56,53,852,47,56,86,10,28,87,98,12,1622,5,4,5,38,15,81,0,0,1},{1765,1399,23,820,931,669,773,826,996,262,97,396,646,1066,554,646,637,448,872,438,174,284,45,302,174,162,129,16,99,163,175,626,101,52,93,12,136,75,95,11,9287,8,2,6,40,23,46,1,2,2},{1771,1732,21,698,931,595,690,821,891,254,86,1314,666,1021,531,553,634,404,177,498,217,267,45,290,181,145,116,17,83,62,51,643,37,52,102,8,33,76,95,9,82,5,6,5,42,12,82,0,0,1},{1723,1100,19,842,954,717,712,839,896,360,95,537,630,868,525,576,615,399,398,353,215,241,45,285,174,160,128,17,92,158,168,462,86,50,104,14,125,73,91,13,88,7,3,6,42,23,41,1,1,2},{1781,1583,13,741,1060,574,751,808,868,329,86,482,746,1004,517,502,597,439,256,1235,208,262,44,274,181,149,103,15,92,50,47,805,40,55,90,8,30,69,92,14,94,5,2,6,42,34,71,0,1,0},{2315,1472,15,706,1231,555,690,822,890,373,94,387,694,934,526,571,554,415,451,3237,144,249,45,268,150,146,113,17,85,151,189,408,96,45,100,9,127,87,101,12,610,7,2,3,42,23,121,2,3,1},{2152,1410,13,677,1430,492,654,860,815,297,85,431,826,1089,518,575,520,428,97,1262,63,292,44,267,187,145,103,14,89,57,67,11,38,46,114,7,30,89,92,10,148,5,2,5,41,18,139,1,0,0},{2577,1406,16,763,1149,569,640,603,646,307,90,324,468,1310,430,543,478,259,549,177,96,200,44,33948,131,114,99,12,85,152,147,25,83,40,134,11,125,78,82,10,113,8,3,4,40,24,708,1,2,1},{2244,1615,15,673,1359,540,625,681,728,373,87,346,543,1182,513,490,561,400,111,187,118,255,44,18858,179,142,101,18,86,50,51,11,37,45,111,6,28,68,101,11,119,5,4,3,39,39,375,1,0,1},{2012,1242,12,792,1145,1588,742,676,762,649,93,467,537,1088,541,577,570,403,434,160,131,273,44,8458,175,157,110,18,98,158,147,0,96,61,122,7,121,72,100,10,-24,9,2,5,38,65,131,1,2,1},{1899,1542,17,735,784,1654,801,811,978,387,84,338,607,1276,561,521,577,438,169,134,105,279,44,5406,195,152,117,17,93,63,52,0,40,56,105,8,31,75,98,12,-18,4,3,5,37,33,24,1,0,0},{2405,1179,13023,858,423,920,706,710,630,418,7977,420,522,794,550,506,541,377,386,126,95,263,94,3137,164,180,408,1903,340,146,134,-1,108,42,61,405,157,64,92,905,-11,306,5,2,37,34,105,1,1,1} };

a = { {1475,1276,10,6245,265,533,666,677,748,426,53,386,662,581,567,373,491,368,206,62,158,226,48,1185,139,352,109,12,81,37,63,0,36,1008,30,3,26,64,103,7,62,1,4,15,11,38,44,0,0,0,0,1,0,0,64,0,1,0,0,26,24,1,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,128,0,1,0,1,1,0,0,0,0,4,1,0,81,25,0,1,0},{1438,1006,12,833,219,569,779,808,853,261,6931,526,674,483,613,428,544,423,401,65,196,246,50,747,175,293,125,12,110,155,189,0,115,53,26,7,149,74,102,1221,19,5,1,23,11,70,74,2,2,1,2,0,2,1,86,3,2,3,3,26,21,2,2,1,1,1,2,2,1,1,1,1,1,0,35,1,1,1,1,1,1,0,133,1,1,1,4,2,0,1,1,0,4,1,2,78,27,9,1,2},{1376,1621,11,749,201,527,753,766,995,205,78,373,679,426,637,385,548,432,273,71,193,273,476,518,201,309,347,1682,486,37,51,0,45,56,19,296,32,83,148,7,27,71,5,12,12,35,127,1,0,1,0,4,1,1,19,0,0,1,1,31,26,1,1,1,0,0,1,1,1,0,0,0,0,0,41,1,0,0,1,1,1,0,132,0,1,0,1,1,0,1,0,0,4,1,0,76,27,0,1,0},{1459,1501,13421,862,227,618,861,817,1075,313,82,472,810,420,646,457,606,476,477,77,159,282,279,402,213,198,112,12,229,186,145,0,141,60,20,8,168,87,112,7,31,417,2,1,13,36,146,0,2,1,3,1,2,0,24,2,1,2,2,32,29,2,2,2,3,1,1,1,1,1,1,1,1,0,48,1,1,1,1,0,3,0,132,1,2,0,4,2,0,1,0,0,4,1,1,75,29,0,1,0},{1612,936,24,889,329,561,877,909,1118,6951,98,445,922,387,666,400,677,510,115,82,188,323,8781,344,217,178,143,24,104,56,52,0,46,69,28,11,36,100,111,19,60,10,7,1,14,32,739,0,0,2,1,1,1,2,20,1,0,1,1,32,33,1,1,1,0,0,1,1,1,0,0,0,1,0,35,1,0,0,1,0,1,0,132,0,1,1,1,1,0,1,0,0,4,2,1,75,31,0,1,0},{1573,1148,17,813,468,565,736,729,927,486,96,414,761,392,527,455,562,409,560,91,259,268,85,320,184,145,115,14,103,150,124,0,112,58,73,9,139,84,95,7,348,9,6,14,16,43,392,1,1,1,0,2,1,0,16,1,1,1,1,30,28,2,1,1,2,1,1,1,1,2,0,1,2,0,60,2,0,0,1,1,2,0,132,1,1,15,2,2,0,1,1,0,2,2,0,75,28,0,1,0},{1874,1144,18,740,605,511,765,789,890,445,94,491,585,691,525,420,521,408,120,244,573,256,59,317,166,143,115,20,83,48,44,0,38,59,55,11,29,67,95,14,1652,8,4,5,18,25,137,5,0,1,1,3,1,1,14,1,0,2,2,27,24,2,1,1,0,0,1,1,1,0,0,0,0,0,38,1,0,0,1,0,1,0,132,0,2,29,1,1,0,1,0,0,4,2,1,76,27,0,1,0},{2068,1224,24,854,697,609,801,834,1004,307,104,507,786,708,554,660,571,436,408,463,280,268,49,315,192,157,120,22,83,141,154,0,105,47,76,18,135,78,101,15,1690,16,3,6,21,44,25,1,0,2,1,1,3,2,20,2,2,3,3,27,39,2,1,2,1,1,1,2,1,2,1,1,1,0,33,1,1,1,1,1,2,0,132,1,1,26,2,1,0,1,0,0,4,2,2,79,28,0,1,0},{1951,1454,14,747,757,513,706,770,963,426,87,385,697,727,555,664,578,451,201,1839,200,276,48,316,185,158,119,17,98,52,50,359,40,51,79,8,29,76,97,11,1472,5,4,5,24,34,110,2,1,2,1,2,1,2,16,0,1,1,1,22,29,2,1,1,0,0,1,1,1,0,0,0,0,0,66,1,0,0,1,0,1,0,132,0,1,15,1,1,0,1,1,0,4,3,1,81,28,0,1,0},{1923,1261,17,818,727,628,691,695,876,314,92,337,690,842,499,672,544,391,413,481,147,242,46,298,186,135,103,14,86,127,177,587,94,45,68,9,116,77,85,11,1196,8,5,4,28,36,83,2,0,1,1,1,1,2,21,1,1,0,0,25,23,2,1,1,1,1,1,1,1,1,0,0,1,0,36,1,1,0,1,1,1,0,131,1,1,1,3,1,0,1,1,0,4,3,2,82,28,0,1,0},{2098,1519,17,727,787,527,751,699,815,372,86,472,738,804,497,562,538,408,168,396,128,254,46,288,164,150,118,15,90,44,65,431,37,47,77,8,29,79,89,12,1969,6,59,5,32,23,35,0,0,1,1,0,1,0,17,0,1,1,1,32,28,2,1,1,0,0,1,1,1,0,0,0,0,0,35,1,0,0,1,0,1,0,126,0,1,0,1,1,0,1,1,0,4,3,1,81,28,0,1,0},{2783,991,17,825,937,679,724,789,898,294,94,286,720,1111,521,589,552,395,457,202,145,255,45,288,171,176,121,17,80,125,155,73,89,52,66,11,117,85,89,11,2036,10,5,4,35,37,34,1,1,2,1,1,1,2,19,1,0,1,1,25,25,2,2,1,1,0,1,1,1,1,1,0,2,0,20,1,1,0,1,1,2,0,130,1,2,0,2,1,0,1,0,0,2,3,2,80,27,0,1,0},{1824,1252,19,798,949,571,819,804,989,306,89,385,777,969,557,563,609,436,107,264,95,277,45,299,182,159,112,17,98,56,53,852,47,56,86,10,28,87,98,12,1622,5,4,5,38,15,81,0,0,1,0,2,0,1,18,1,1,1,1,24,25,2,1,1,0,0,1,1,1,0,0,0,0,0,35,1,0,0,1,0,1,0,134,0,1,5,1,1,0,0,1,0,4,4,1,82,30,0,1,0},{1765,1399,23,820,931,669,773,826,996,262,97,396,646,1066,554,646,637,448,872,438,174,284,45,302,174,162,129,16,99,163,175,626,101,52,93,12,136,75,95,11,9287,8,2,6,40,23,46,1,2,2,1,1,2,1,13,0,3,2,2,34,25,2,2,2,2,1,1,1,1,1,1,0,2,0,34,2,1,0,1,1,2,0,98,1,2,0,1,2,0,1,0,0,4,4,2,82,29,0,1,0},{1771,1732,21,698,931,595,690,821,891,254,86,1314,666,1021,531,553,634,404,177,498,217,267,45,290,181,145,116,17,83,62,51,643,37,52,102,8,33,76,95,9,82,5,6,5,42,12,82,0,0,1,0,3,0,0,17,1,1,2,2,30,29,2,1,1,0,0,1,1,1,0,0,0,0,0,27,1,0,0,1,0,1,0,121,0,2,0,1,1,0,0,1,0,4,4,1,83,28,0,1,0},{1723,1100,19,842,954,717,712,839,896,360,95,537,630,868,525,576,615,399,398,353,215,241,45,285,174,160,128,17,92,158,168,462,86,50,104,14,125,73,91,13,88,7,3,6,42,23,41,1,1,2,1,1,2,0,18,1,2,2,2,44,28,2,2,1,0,1,1,1,1,1,0,0,1,0,33,2,1,0,1,0,3,0,135,1,2,0,3,2,0,1,1,0,4,5,1,83,28,0,1,0},{1781,1583,13,741,1060,574,751,808,868,329,86,482,746,1004,517,502,597,439,256,1235,208,262,44,274,181,149,103,15,92,50,47,805,40,55,90,8,30,69,92,14,94,5,2,6,42,34,71,0,1,0,0,3,1,1,16,0,1,1,1,24,25,2,1,1,0,0,1,1,1,0,0,0,0,0,33,1,0,0,1,0,1,0,131,0,2,0,1,1,0,0,0,0,4,5,1,83,28,0,1,0},{2315,1472,15,706,1231,555,690,822,890,373,94,387,694,934,526,571,554,415,451,3237,144,249,45,268,150,146,113,17,85,151,189,408,96,45,100,9,127,87,101,12,610,7,2,3,42,23,121,2,3,1,1,2,1,1,18,3,0,3,3,28,26,2,1,1,0,0,1,1,1,0,0,0,1,0,10,2,1,0,1,1,2,0,77,0,2,0,2,2,0,1,0,0,2,5,2,82,28,0,1,0},{2152,1410,13,677,1430,492,654,860,815,297,85,431,826,1089,518,575,520,428,97,1262,63,292,44,267,187,145,103,14,89,57,67,11,38,46,114,7,30,89,92,10,148,5,2,5,41,18,139,1,0,0,0,4,0,0,19,0,1,1,1,24,24,2,1,1,0,0,1,1,1,0,0,0,0,0,30,1,0,0,1,0,1,0,71,0,2,0,1,1,0,0,0,0,4,5,1,83,27,0,1,0},{2577,1406,16,763,1149,569,640,603,646,307,90,324,468,1310,430,543,478,259,549,177,96,200,44,33948,131,114,99,12,85,152,147,25,83,40,134,11,125,78,82,10,113,8,3,4,40,24,708,1,2,1,2,6,1,1,21,2,1,3,3,22,26,2,1,1,1,0,1,1,1,0,0,0,2,0,5,2,0,0,1,1,2,0,125,1,1,0,3,2,0,1,1,0,4,5,2,83,29,0,1,0},{2244,1615,15,673,1359,540,625,681,728,373,87,346,543,1182,513,490,561,400,111,187,118,255,44,18858,179,142,101,18,86,50,51,11,37,45,111,6,28,68,101,11,119,5,4,3,39,39,375,1,0,1,0,20,0,1,29,1,0,2,2,27,27,2,1,0,0,0,1,1,1,0,0,0,0,0,31,1,0,0,1,0,1,0,114,0,2,0,1,1,0,0,0,0,4,6,1,83,28,0,1,0},{2012,1242,12,792,1145,1588,742,676,762,649,93,467,537,1088,541,577,570,403,434,160,131,273,44,8458,175,157,110,18,98,158,147,0,96,61,122,7,121,72,100,10,24,9,2,5,38,65,131,1,2,1,1,30,2,1,37,1,1,0,0,37,40,2,1,1,2,0,1,1,1,0,0,0,1,0,29,1,1,0,1,0,2,0,121,1,1,0,2,2,0,1,0,0,4,6,1,82,28,0,1,0},{1899,1542,17,735,784,1654,801,811,978,387,84,338,607,1276,561,521,577,438,169,134,105,279,44,5406,195,152,117,17,93,63,52,0,40,56,105,8,31,75,98,12,18,4,3,5,37,33,24,1,0,0,0,10,0,1,116,1,0,2,2,27,30,2,1,0,0,0,1,1,1,0,0,0,0,0,7,1,0,0,1,0,1,0,134,0,1,0,1,1,0,0,0,0,4,6,1,80,28,0,1,0},{2405,1179,13023,858,423,920,706,710,630,418,7977,420,522,794,550,506,541,377,386,126,95,263,94,3137,164,180,408,1903,340,146,134,1,108,42,61,405,157,64,92,905,11,306,5,2,37,34,105,1,1,1,0,0,1,1,63,1,2,3,3,24,23,2,1,1,1,0,1,1,1,0,0,0,1,0,18,1,1,0,1,0,1,0,117,1,2,0,2,2,0,1,0,0,4,7,1,79,28,0,1,0} };

/\*a.push\_back({});

for (int i = 0; i < a[0].size(); i++) {

a[a.size() - 1].push\_back(0);

}\*/

/\*for (auto& vec : a) {

int i = 9;

while(i--)

vec.pop\_back();

}\*/

b = { {2.183,0.696,0.032,0.548,0.071,0.074,1.526,0.113},{1.497,0.344,0.101,0.515,0.006,0.085,1.711,0.109},{1.473,0.677,0.021,0.414,0.026,4.948,1.4,0.106},{1.201,0.394,0.124,0.829,0.058,2.883,2.03,0.639},{1.303,1.039,0.043,0.573,0.084,5.029,2.606,0.699},{1.354,0.857,0.085,0.576,0.114,0.425,2.011,0.607},{1.532,0.798,0.015,0.521,0.171,0.223,1.903,1.204},{2.31,0.491,0.071,0.855,1.933,0.138,1.983,1.233},{2.72,1.036,0.024,0.657,1.9,0.099,2.144,0.728},{2.553,0.704,0.11,1.134,2.357,0.06,2.077,0.973},{2.299,0.569,0.017,0.897,2.642,0.051,1.933,0.686},{2.124,0.79,0.074,0.453,0.575,0.041,3.283,0.902},{2.158,0.563,0.329,0.421,2.189,0.036,3.026,0.81},{2.296,0.488,0.079,0.423,2.377,0.036,2.277,0.709},{2.314,1.158,0.033,0.427,2.178,0.035,1.922,3.753},{2.133,0.742,0.119,0.677,2.313,0.034,1.919,5.065},{1.939,0.622,0.029,0.568,1.993,0.031,1.432,0.756},{1.965,0.857,0.072,0.431,1.406,0.035,1.516,0.417},{2.268,0.59,0.015,0.763,0.943,0.032,1.409,0.432},{1.981,0.494,0.087,0.523,0.55,0.032,1.476,0.45},{1.911,0.603,0.062,0.515,0.557,0.031,1.644,0.643},{2.014,0.604,0.132,0.46,0.585,0.029,1.603,0.494},{1.999,0.564,0.029,0.748,0.681,0.029,1.692,0.705},{2.109,0.847,0.046,0.573,0.61,0.049,1.747,0.966} };

/\*b.push\_back({});

for (int i = 0; i < b[0].size(); i++) {

b[b.size() - 1].push\_back(0);

}\*/

a\_ = a;

allocat\_ans = vector<vector<int>>(a.size(),vector<int>(a[0].size(),0));

/\*for (int i = 0; i < a.size(); i++) {

for (int j = 0; j < a[i].size(); j++) {

a[i][j] \*= 1.1;

}

}\*/

cumsum = vector<double>(weeks, 0);

double expect = 0;

double lan = 0;

for (int j = 0; j < a[0].size(); j++)cumsum[0] += a[0][j]/actual\_consume[kind\_dict[j]-1];

for (int i = 1; i < a.size(); i++) {

double sum = 0;

for (int j = 0; j < a[i].size(); j++) {

sum += a[i][j]/actual\_consume[kind\_dict[j]-1];

}

cumsum[i] = cumsum[i - 1] + sum;

}

for (int i = weeks-1; i >= 0; i--) {

double sum = 0;

for (int j = 0; j < a[i].size(); j++) {

sum += a[i][j] / actual\_consume[kind\_dict[j] - 1];

}

//cout << lan << endl;

if (sum-capacity\_need > 0) {

if (lan < 0) {

if (sum - capacity\_need + lan > 0) {

expect += sum - capacity\_need + lan;

lan = 0;

}

else {

lan += sum - capacity\_need;

}

}

else {

expect += sum - capacity\_need;;

}

}

else {

lan += sum - capacity\_need;

}

}

cout << "不计损耗、计时序拓扑截尾后的产能上限为" << capacity\_need+lan/24<<endl;

for (auto& c : cumsum) {

if (c / capacity\_need >= useful\_bound.size() + 1)useful\_bound.push\_back(true);

else useful\_bound.push\_back(false);

//cout << c / capacity\_need <<" "<<useful\_bound.size()<<endl;

}

}

vector<pair<int, pair<int, double>>> optical\_trans\_order(int weeks,int nums = 4) {

vector<pair<int, pair<int, double>>> opt\_set;//triple->(week,no.,rate)

for (int k = 0; k < weeks; k++) {

vector<pair<int, double>> tup;

for (int l = 0; l < b[k].size(); l++) {

tup.push\_back(make\_pair(l,b[k][l]));

}

sort(tup.begin(), tup.end(), [](auto& x, auto& y) {return x.second < y.second; });

//放入损耗率最小的前四个即可

for (int i = 0; i < nums; i++) {

opt\_set.push\_back(make\_pair(k, tup[i]));

}

}

sort(opt\_set.begin(), opt\_set.end(), [](auto& x, auto& y) {return x.second.second < y.second.second; });

return opt\_set;

}

bool execute() {

auto steps = optical\_trans\_order(weeks,4);

for (auto& step : steps) {

int week = step.first;//获取周数

int no = step.second.first;//获取仓库号

for (int kind = 1; kind < 4; kind++) {

for (int i = week; i >= 0; i--) {//倒序依次装配A B C货物

for (int pro = 0; pro < a[week].size(); pro++) {//其实可以优化一下，减少外部碎片，这里暂时先不管了

if (kind\_dict[pro] != kind||a[i][pro]==0)continue;

if (weekly\_actual\_capacity[week] >= capacity\_need-1)continue;

//检验边界情况是否越出

bool drop = false;

int count = 0;

double all = 0;

for (int aloc\_i = 0; aloc\_i < i; aloc\_i++) {

all += capacity\_need - weekly\_actual\_capacity[aloc\_i];

}

for (int aloc\_i = i; aloc\_i < week; aloc\_i++) {

all += capacity\_need - weekly\_actual\_capacity[aloc\_i];

if (useful\_bound[aloc\_i]) {//检测是否越界了

if (all > cumsum[aloc\_i]) {

drop = true;

break;

}

}

}

//if(week>=1)

/\*for (int aloc\_i = i; aloc\_i <week; aloc\_i++) {

all += capacity\_need - weekly\_actual\_capacity[aloc\_i];

if (!useful\_bound[aloc\_i])break;

if (all > cumsum[aloc\_i]&&useful\_bound[aloc\_i]) {

drop = true;

break;

//count++;

}

}\*/

//if (count == week - i-1||count == 0)drop = false;

//else drop = true;

//检测有没有库存超限，超限情况下应当另外讨论，这里为了简化模型选择直接放弃此次分配

int minist = 6000;

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

minist = min(block[aloc\_i][no][3], minist);

if (block[aloc\_i][no][3] <=0){//< a[i][pro]) {

drop = true;

break;

}

}

if (drop) break;

minist = min(minist, (int)((capacity\_need - weekly\_actual\_capacity[week])\*actual\_consume[kind-1]));

minist = max(minist, 0);

//必要的时候分割转运操作避免爆仓

if (minist > 0 && minist < a[i][pro]) {

//cout << a[i][pro]<<" ";

a[i][pro] -= minist;

int tmp = minist;

minist = a[i][pro];

a[i][pro] = tmp;

//cout << a[i][pro] << endl;

}

else minist = 0;

//cout << a[i][pro] << endl;

//通过检测则放入

weekly\_actual\_capacity[week] += (1. - 0.01\*b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1])+0.5;//(1-损耗率）\*总量/单位消耗

ave += (1. - 0.01 \* b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1]);

//cout << weekly\_actual\_capacity[week] << endl;

//在[i,week]期间，将货物存放到no号仓库内，更新时空图和仓储分区与实际产能,校验累积和边界

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

cumsum[aloc\_i] -= a[i][pro] / (actual\_consume[kind - 1]);

block[aloc\_i][no][kind-1] += a[i][pro];

block[aloc\_i][no][3] -= a[i][pro];

allocate\_status\_map[aloc\_i][no][kind - 1].push(make\_pair(make\_pair(i, pro),a[i][pro]));

//cout << i << " " << pro << " " << a\_[i][pro] << endl;

}

allocat\_ans[i][pro] += a[i][pro];

//清空供货商相关存量

a[i][pro] = minist;

}

}

}

}//with\_bound

for (int i = 0; i < useful\_bound.size(); i++) {

if (weekly\_actual\_capacity[i] >= capacity\_need - 1) {

useful\_bound[i] = true;

}

else {

useful\_bound[i] = false;

//cout << i << endl;

}

}

capacity\_need \*= 2;//获取超出的材料

steps = optical\_trans\_order(weeks, 8);

//sort(steps.begin(), steps.end(), [](auto& x, auto& y) {return (x.first < y.first); });

for (auto& step : steps) {

int week = step.first;//获取周数

int no = step.second.first;//获取仓库号

for (int i = week; i >= 0; i--) {

for (int kind = 1; kind < 4; kind++) {//倒序依次装配A B C货物

for (int pro = 0; pro < a[week].size(); pro++) {//其实可以优化一下，减少外部碎片，这里暂时先不管了

if (kind\_dict[pro] != kind || a[i][pro] == 0)continue;

if (useful\_bound[week])continue;

bool drop = false;

//检测是否有必要扩

//检测有没有库存超限，超限情况下应当另外讨论，这里为了简化模型选择直接放弃此次分配

int minist = 6000;

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

minist = min(block[aloc\_i][no][3], minist);

if (block[aloc\_i][no][3] <= 0) {//< a[i][pro]) {

drop = true;

break;

}

}

//cout << minist << endl;

if (drop) break;

minist = min(minist, (int)((capacity\_need - weekly\_actual\_capacity[week]) \* actual\_consume[kind - 1]));

minist = max(minist, 0);

//必要的时候分割转运操作避免爆仓

if (minist > 0 && minist < a[i][pro]) {

//cout << a[i][pro]<<" ";

a[i][pro] -= minist;

int tmp = minist;

minist = a[i][pro];

a[i][pro] = tmp;

//cout << a[i][pro] << endl;

}

else minist = 0;

//cout << a[i][pro] << endl;

//通过检测则放入

weekly\_actual\_capacity[week] += (1. - 0.01 \* b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1])+0.5;//(1-损耗率）\*总量/单位消耗

ave += (1. - 0.01 \* b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1]);

if (weekly\_actual\_capacity[week] >= capacity\_need / 2 - 1)useful\_bound[week] = true;

//在[i,week]期间，将货物存放到no号仓库内，更新时空图和仓储分区与实际产能,校验累积和边界

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

cumsum[aloc\_i] -= a[i][pro] / (actual\_consume[kind - 1]);

block[aloc\_i][no][kind - 1] += a[i][pro];

block[aloc\_i][no][3] -= a[i][pro];

allocate\_status\_map[aloc\_i][no][kind - 1].push(make\_pair(make\_pair(i, pro), a[i][pro]));

//cout << i << " " << pro << " " << a\_[i][pro] << endl;

}

allocat\_ans[i][pro] += a[i][pro];

//清空供货商相关存量

a[i][pro] = minist;

}

}

}

}//without bound

cout << "周产能:";

int sum = 0;

for (auto& e : weekly\_actual\_capacity) {

cout << e << " ";

sum += e;

}

cout << endl;

cout << ave/24 << endl;

cout << "转运决策如下：";

sort(steps.begin(), steps.end(), [](auto& x, auto& y) {return x.first < y.first; });

//cout << "week1:" << endl;

int nums = 8;

for (int i = 0; i < steps.size(); i++) {

if (i % nums == 0)cout <<"第"<<steps[i].first<<"周"<< endl;

if (block[i/nums][steps[i].second.first][3] == 6000)continue;

cout <<"仓库" << steps[i].second.first <<" 损耗率"<< steps[i].second.second<<" 运送"<< endl;

cout << "仓储分区情况：" << endl;

cout << "A:" <<block[i/nums][steps[i].second.first][0]<< " B:"<< block[i / nums][steps[i].second.first][1] << " C:" << block[i / nums][steps[i].second.first][2] << " empty:" << block[i / nums][steps[i].second.first][3] << endl;

cout << "具体策略如下：" << endl;

int e = steps[i].second.first;

for (int j = 0; j < 3; j++) {

cout << "材料种类" << j << endl;

while (!allocate\_status\_map[i / nums][e][j].empty()) {

auto& tmp = allocate\_status\_map[i / nums][e][j].top();

allocate\_status\_map[i / nums][e][j].pop();

cout << "存入周:" << tmp.first.first << "\t商家:" << tmp.first.second << "\t 数量" << tmp.second <<'\t'<< endl;

//allocat\_ans[tmp.first.first][tmp.first.second] += tmp.second;

}

}

}

cout << "订货情况：（每行 为一个商家的数据）"<<endl;

for (int i = 0; i < a[0].size(); i++) {

for (int j = 0; j < a.size(); j++) {

cout << allocat\_ans[j][i]<< '\t';

}

cout << endl;

}

cout << endl;

cout << "材料空闲情况" << endl;

for (int i = 0; i < a[0].size(); i++) {

for (int j = 0; j < a.size(); j++) {

cout << (-allocat\_ans[j][i] + a\_[j][i])<< '\t';

}

cout << endl;

}

return true;

}

};

int main(void) {

Solution s = Solution();

s.execute();

char ch;

cin >> ch;

return 0;

}

以下为第三问的C++代码部分：

#include<iostream>

#include<vector>

#include<queue>

#include<map>

using namespace std;

class Solution {

private:

vector<double> d;//表示1m^3货物的制造产能

vector<double> e;//表示订购单位j种货物的价格

vector<double> m;//表示k周总供货量

vector<double> z;//表示k周总损耗

vector<double> w;//表示k周总成本

vector<double> u\_1;//表示k周采购成本

vector<double> u\_2;//表示k周运输成本

vector<double> u\_3;//表示k周存储成本

double f0, e0, g0;//运输、存储、损耗

//仓库货物时空分布图

int weeks;

double capacity\_need;

vector<vector<int>> a;//表示在第k周第i家供货商的供货量

vector<vector<int>> a\_;

vector<int> kind\_dict;//供货商商品种类映射 A-1,B-2,C-3

vector<double> actual\_consume;//A-0.6,B-0.66,C-0.72

vector<vector<vector<priority\_queue<pair<pair<int, int>,int>, vector<pair<pair<int, int>,int>>>>>> allocate\_status\_map;//[周][仓库]->(（周数，商家）,数量)

vector<vector<vector<int>>> block;//[周][仓库号]仓储分区(A-B-C-EMPTY)

vector<int> weekly\_actual\_capacity;//实际产能--按周记录

vector<vector<double>> b;//表示在k周第L家转运商的转运损耗率

vector<double> cumsum;//周累积和边界检测，需要在保证每周产能的前提下减少购买成本和贮存成本

vector<bool> useful\_bound;

vector<vector<int>> allocat\_ans;

double ave;

public:

Solution() {

ave = 0;

weeks = 24;

capacity\_need = 28200.\*2;

//kind\_dict = { 3,3,3,3,2,3,3,2,1,1,1,2,3,1,2,2,1,2,3,1,1,1,1,3,1,1,3,2,3,3,1,3,1,1,3,1,2,1,3,1,2,2,2,3,2,3,1,1,1,2 };//2

kind\_dict = { 3,3,3,3,2,3,3,2,1,1,1,2,3,1,2,2,1,2,3,1,1,1,1,3,1,1,3,2,3,3,1,3,1,1,3,1,2,1,3,1,2,2,2,3,2,3,1,1,1,2,

1,1,2,1,1,1,1,2,2,1,1,3,1,1,2,3,2,2,2,1,3,3,1,1,2,1,3,1,3,2,1,1,3,3,2,2,1,1,3,2,3,3,3,1,1,1,3,3,2,3};

actual\_consume = { 0.6,0.66,0.72 };

block = vector<vector<vector<int>>>(weeks, vector<vector<int>>(50, vector<int>(4, 0)));

for (int i = 0; i < block.size(); i++) {

for (int j = 0; j < block[i].size(); j++) {

block[i][j][3] = 6000;

}

}

weekly\_actual\_capacity = vector<int>(weeks, 0);

allocate\_status\_map = vector<vector<vector<priority\_queue<pair<pair<int, int>, int>, vector<pair<pair<int, int>, int>>>>>>(weeks, vector<vector<priority\_queue<pair<pair<int, int>, int>, vector<pair<pair<int, int>, int>>>>>(8, vector<priority\_queue<pair<pair<int, int>, int>, vector<pair<pair<int, int>, int>>>>(3)));

//a = { {1475,1276,10,6245,265,533,666,677,748,426,53,386,662,581,567,373,491,368,206,62,158,226,48,1185,139,352,109,12,81,37,63,0,36,1008,30,3,26,64,103,7,62,1,4,15,11,38,44,0,0,0},{1438,1006,12,833,219,569,779,808,853,261,6931,526,674,483,613,428,544,423,401,65,196,246,50,747,175,293,125,12,110,155,189,0,115,53,26,7,149,74,102,1221,19,5,1,23,11,70,74,2,2,1},{1376,1621,11,749,201,527,753,766,995,205,78,373,679,426,637,385,548,432,273,71,193,273,476,518,201,309,347,1682,486,37,51,0,45,56,19,296,32,83,148,7,27,71,5,12,12,35,127,1,0,1},{1459,1501,13421,862,227,618,861,817,1075,313,82,472,810,420,646,457,606,476,477,77,159,282,279,402,213,198,112,12,229,186,145,0,141,60,20,8,168,87,112,7,31,417,2,1,13,36,146,0,2,1},{1612,936,24,889,329,561,877,909,1118,6951,98,445,922,387,666,400,677,510,115,82,188,323,8781,344,217,178,143,24,104,56,52,0,46,69,28,11,36,100,111,19,60,10,7,1,14,32,739,0,0,2},{1573,1148,17,813,468,565,736,729,927,486,96,414,761,392,527,455,562,409,560,91,259,268,85,320,184,145,115,14,103,150,124,0,112,58,73,9,139,84,95,7,348,9,6,14,16,43,392,1,1,1},{1874,1144,18,740,605,511,765,789,890,445,94,491,585,691,525,420,521,408,120,244,573,256,59,317,166,143,115,20,83,48,44,0,38,59,55,11,29,67,95,14,1652,8,4,5,18,25,137,5,0,1},{2068,1224,24,854,697,609,801,834,1004,307,104,507,786,708,554,660,571,436,408,463,280,268,49,315,192,157,120,22,83,141,154,0,105,47,76,18,135,78,101,15,1690,16,3,6,21,44,25,1,0,2},{1951,1454,14,747,757,513,706,770,963,426,87,385,697,727,555,664,578,451,201,1839,200,276,48,316,185,158,119,17,98,52,50,359,40,51,79,8,29,76,97,11,1472,5,4,5,24,34,110,2,1,2},{1923,1261,17,818,727,628,691,695,876,314,92,337,690,842,499,672,544,391,413,481,147,242,46,298,186,135,103,14,86,127,177,587,94,45,68,9,116,77,85,11,1196,8,5,4,28,36,83,2,0,1},{2098,1519,17,727,787,527,751,699,815,372,86,472,738,804,497,562,538,408,168,396,128,254,46,288,164,150,118,15,90,44,65,431,37,47,77,8,29,79,89,12,1969,6,59,5,32,23,35,0,0,1},{2783,991,17,825,937,679,724,789,898,294,94,286,720,1111,521,589,552,395,457,202,145,255,45,288,171,176,121,17,80,125,155,73,89,52,66,11,117,85,89,11,2036,10,5,4,35,37,34,1,1,2},{1824,1252,19,798,949,571,819,804,989,306,89,385,777,969,557,563,609,436,107,264,95,277,45,299,182,159,112,17,98,56,53,852,47,56,86,10,28,87,98,12,1622,5,4,5,38,15,81,0,0,1},{1765,1399,23,820,931,669,773,826,996,262,97,396,646,1066,554,646,637,448,872,438,174,284,45,302,174,162,129,16,99,163,175,626,101,52,93,12,136,75,95,11,9287,8,2,6,40,23,46,1,2,2},{1771,1732,21,698,931,595,690,821,891,254,86,1314,666,1021,531,553,634,404,177,498,217,267,45,290,181,145,116,17,83,62,51,643,37,52,102,8,33,76,95,9,82,5,6,5,42,12,82,0,0,1},{1723,1100,19,842,954,717,712,839,896,360,95,537,630,868,525,576,615,399,398,353,215,241,45,285,174,160,128,17,92,158,168,462,86,50,104,14,125,73,91,13,88,7,3,6,42,23,41,1,1,2},{1781,1583,13,741,1060,574,751,808,868,329,86,482,746,1004,517,502,597,439,256,1235,208,262,44,274,181,149,103,15,92,50,47,805,40,55,90,8,30,69,92,14,94,5,2,6,42,34,71,0,1,0},{2315,1472,15,706,1231,555,690,822,890,373,94,387,694,934,526,571,554,415,451,3237,144,249,45,268,150,146,113,17,85,151,189,408,96,45,100,9,127,87,101,12,610,7,2,3,42,23,121,2,3,1},{2152,1410,13,677,1430,492,654,860,815,297,85,431,826,1089,518,575,520,428,97,1262,63,292,44,267,187,145,103,14,89,57,67,11,38,46,114,7,30,89,92,10,148,5,2,5,41,18,139,1,0,0},{2577,1406,16,763,1149,569,640,603,646,307,90,324,468,1310,430,543,478,259,549,177,96,200,44,33948,131,114,99,12,85,152,147,25,83,40,134,11,125,78,82,10,113,8,3,4,40,24,708,1,2,1},{2244,1615,15,673,1359,540,625,681,728,373,87,346,543,1182,513,490,561,400,111,187,118,255,44,18858,179,142,101,18,86,50,51,11,37,45,111,6,28,68,101,11,119,5,4,3,39,39,375,1,0,1},{2012,1242,12,792,1145,1588,742,676,762,649,93,467,537,1088,541,577,570,403,434,160,131,273,44,8458,175,157,110,18,98,158,147,0,96,61,122,7,121,72,100,10,-24,9,2,5,38,65,131,1,2,1},{1899,1542,17,735,784,1654,801,811,978,387,84,338,607,1276,561,521,577,438,169,134,105,279,44,5406,195,152,117,17,93,63,52,0,40,56,105,8,31,75,98,12,-18,4,3,5,37,33,24,1,0,0},{2405,1179,13023,858,423,920,706,710,630,418,7977,420,522,794,550,506,541,377,386,126,95,263,94,3137,164,180,408,1903,340,146,134,-1,108,42,61,405,157,64,92,905,-11,306,5,2,37,34,105,1,1,1} };

a = { {1475,1276,10,6245,265,533,666,677,748,426,53,386,662,581,567,373,491,368,206,62,158,226,48,1185,139,352,109,12,81,37,63,0,36,1008,30,3,26,64,103,7,62,1,4,15,11,38,44,0,0,0,0,1,0,0,64,0,1,0,0,26,24,1,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,128,0,1,0,1,1,0,0,0,0,4,1,0,81,25,0,1,0},{1438,1006,12,833,219,569,779,808,853,261,6931,526,674,483,613,428,544,423,401,65,196,246,50,747,175,293,125,12,110,155,189,0,115,53,26,7,149,74,102,1221,19,5,1,23,11,70,74,2,2,1,2,0,2,1,86,3,2,3,3,26,21,2,2,1,1,1,2,2,1,1,1,1,1,0,35,1,1,1,1,1,1,0,133,1,1,1,4,2,0,1,1,0,4,1,2,78,27,9,1,2},{1376,1621,11,749,201,527,753,766,995,205,78,373,679,426,637,385,548,432,273,71,193,273,476,518,201,309,347,1682,486,37,51,0,45,56,19,296,32,83,148,7,27,71,5,12,12,35,127,1,0,1,0,4,1,1,19,0,0,1,1,31,26,1,1,1,0,0,1,1,1,0,0,0,0,0,41,1,0,0,1,1,1,0,132,0,1,0,1,1,0,1,0,0,4,1,0,76,27,0,1,0},{1459,1501,13421,862,227,618,861,817,1075,313,82,472,810,420,646,457,606,476,477,77,159,282,279,402,213,198,112,12,229,186,145,0,141,60,20,8,168,87,112,7,31,417,2,1,13,36,146,0,2,1,3,1,2,0,24,2,1,2,2,32,29,2,2,2,3,1,1,1,1,1,1,1,1,0,48,1,1,1,1,0,3,0,132,1,2,0,4,2,0,1,0,0,4,1,1,75,29,0,1,0},{1612,936,24,889,329,561,877,909,1118,6951,98,445,922,387,666,400,677,510,115,82,188,323,8781,344,217,178,143,24,104,56,52,0,46,69,28,11,36,100,111,19,60,10,7,1,14,32,739,0,0,2,1,1,1,2,20,1,0,1,1,32,33,1,1,1,0,0,1,1,1,0,0,0,1,0,35,1,0,0,1,0,1,0,132,0,1,1,1,1,0,1,0,0,4,2,1,75,31,0,1,0},{1573,1148,17,813,468,565,736,729,927,486,96,414,761,392,527,455,562,409,560,91,259,268,85,320,184,145,115,14,103,150,124,0,112,58,73,9,139,84,95,7,348,9,6,14,16,43,392,1,1,1,0,2,1,0,16,1,1,1,1,30,28,2,1,1,2,1,1,1,1,2,0,1,2,0,60,2,0,0,1,1,2,0,132,1,1,15,2,2,0,1,1,0,2,2,0,75,28,0,1,0},{1874,1144,18,740,605,511,765,789,890,445,94,491,585,691,525,420,521,408,120,244,573,256,59,317,166,143,115,20,83,48,44,0,38,59,55,11,29,67,95,14,1652,8,4,5,18,25,137,5,0,1,1,3,1,1,14,1,0,2,2,27,24,2,1,1,0,0,1,1,1,0,0,0,0,0,38,1,0,0,1,0,1,0,132,0,2,29,1,1,0,1,0,0,4,2,1,76,27,0,1,0},{2068,1224,24,854,697,609,801,834,1004,307,104,507,786,708,554,660,571,436,408,463,280,268,49,315,192,157,120,22,83,141,154,0,105,47,76,18,135,78,101,15,1690,16,3,6,21,44,25,1,0,2,1,1,3,2,20,2,2,3,3,27,39,2,1,2,1,1,1,2,1,2,1,1,1,0,33,1,1,1,1,1,2,0,132,1,1,26,2,1,0,1,0,0,4,2,2,79,28,0,1,0},{1951,1454,14,747,757,513,706,770,963,426,87,385,697,727,555,664,578,451,201,1839,200,276,48,316,185,158,119,17,98,52,50,359,40,51,79,8,29,76,97,11,1472,5,4,5,24,34,110,2,1,2,1,2,1,2,16,0,1,1,1,22,29,2,1,1,0,0,1,1,1,0,0,0,0,0,66,1,0,0,1,0,1,0,132,0,1,15,1,1,0,1,1,0,4,3,1,81,28,0,1,0},{1923,1261,17,818,727,628,691,695,876,314,92,337,690,842,499,672,544,391,413,481,147,242,46,298,186,135,103,14,86,127,177,587,94,45,68,9,116,77,85,11,1196,8,5,4,28,36,83,2,0,1,1,1,1,2,21,1,1,0,0,25,23,2,1,1,1,1,1,1,1,1,0,0,1,0,36,1,1,0,1,1,1,0,131,1,1,1,3,1,0,1,1,0,4,3,2,82,28,0,1,0},{2098,1519,17,727,787,527,751,699,815,372,86,472,738,804,497,562,538,408,168,396,128,254,46,288,164,150,118,15,90,44,65,431,37,47,77,8,29,79,89,12,1969,6,59,5,32,23,35,0,0,1,1,0,1,0,17,0,1,1,1,32,28,2,1,1,0,0,1,1,1,0,0,0,0,0,35,1,0,0,1,0,1,0,126,0,1,0,1,1,0,1,1,0,4,3,1,81,28,0,1,0},{2783,991,17,825,937,679,724,789,898,294,94,286,720,1111,521,589,552,395,457,202,145,255,45,288,171,176,121,17,80,125,155,73,89,52,66,11,117,85,89,11,2036,10,5,4,35,37,34,1,1,2,1,1,1,2,19,1,0,1,1,25,25,2,2,1,1,0,1,1,1,1,1,0,2,0,20,1,1,0,1,1,2,0,130,1,2,0,2,1,0,1,0,0,2,3,2,80,27,0,1,0},{1824,1252,19,798,949,571,819,804,989,306,89,385,777,969,557,563,609,436,107,264,95,277,45,299,182,159,112,17,98,56,53,852,47,56,86,10,28,87,98,12,1622,5,4,5,38,15,81,0,0,1,0,2,0,1,18,1,1,1,1,24,25,2,1,1,0,0,1,1,1,0,0,0,0,0,35,1,0,0,1,0,1,0,134,0,1,5,1,1,0,0,1,0,4,4,1,82,30,0,1,0},{1765,1399,23,820,931,669,773,826,996,262,97,396,646,1066,554,646,637,448,872,438,174,284,45,302,174,162,129,16,99,163,175,626,101,52,93,12,136,75,95,11,9287,8,2,6,40,23,46,1,2,2,1,1,2,1,13,0,3,2,2,34,25,2,2,2,2,1,1,1,1,1,1,0,2,0,34,2,1,0,1,1,2,0,98,1,2,0,1,2,0,1,0,0,4,4,2,82,29,0,1,0},{1771,1732,21,698,931,595,690,821,891,254,86,1314,666,1021,531,553,634,404,177,498,217,267,45,290,181,145,116,17,83,62,51,643,37,52,102,8,33,76,95,9,82,5,6,5,42,12,82,0,0,1,0,3,0,0,17,1,1,2,2,30,29,2,1,1,0,0,1,1,1,0,0,0,0,0,27,1,0,0,1,0,1,0,121,0,2,0,1,1,0,0,1,0,4,4,1,83,28,0,1,0},{1723,1100,19,842,954,717,712,839,896,360,95,537,630,868,525,576,615,399,398,353,215,241,45,285,174,160,128,17,92,158,168,462,86,50,104,14,125,73,91,13,88,7,3,6,42,23,41,1,1,2,1,1,2,0,18,1,2,2,2,44,28,2,2,1,0,1,1,1,1,1,0,0,1,0,33,2,1,0,1,0,3,0,135,1,2,0,3,2,0,1,1,0,4,5,1,83,28,0,1,0},{1781,1583,13,741,1060,574,751,808,868,329,86,482,746,1004,517,502,597,439,256,1235,208,262,44,274,181,149,103,15,92,50,47,805,40,55,90,8,30,69,92,14,94,5,2,6,42,34,71,0,1,0,0,3,1,1,16,0,1,1,1,24,25,2,1,1,0,0,1,1,1,0,0,0,0,0,33,1,0,0,1,0,1,0,131,0,2,0,1,1,0,0,0,0,4,5,1,83,28,0,1,0},{2315,1472,15,706,1231,555,690,822,890,373,94,387,694,934,526,571,554,415,451,3237,144,249,45,268,150,146,113,17,85,151,189,408,96,45,100,9,127,87,101,12,610,7,2,3,42,23,121,2,3,1,1,2,1,1,18,3,0,3,3,28,26,2,1,1,0,0,1,1,1,0,0,0,1,0,10,2,1,0,1,1,2,0,77,0,2,0,2,2,0,1,0,0,2,5,2,82,28,0,1,0},{2152,1410,13,677,1430,492,654,860,815,297,85,431,826,1089,518,575,520,428,97,1262,63,292,44,267,187,145,103,14,89,57,67,11,38,46,114,7,30,89,92,10,148,5,2,5,41,18,139,1,0,0,0,4,0,0,19,0,1,1,1,24,24,2,1,1,0,0,1,1,1,0,0,0,0,0,30,1,0,0,1,0,1,0,71,0,2,0,1,1,0,0,0,0,4,5,1,83,27,0,1,0},{2577,1406,16,763,1149,569,640,603,646,307,90,324,468,1310,430,543,478,259,549,177,96,200,44,33948,131,114,99,12,85,152,147,25,83,40,134,11,125,78,82,10,113,8,3,4,40,24,708,1,2,1,2,6,1,1,21,2,1,3,3,22,26,2,1,1,1,0,1,1,1,0,0,0,2,0,5,2,0,0,1,1,2,0,125,1,1,0,3,2,0,1,1,0,4,5,2,83,29,0,1,0},{2244,1615,15,673,1359,540,625,681,728,373,87,346,543,1182,513,490,561,400,111,187,118,255,44,18858,179,142,101,18,86,50,51,11,37,45,111,6,28,68,101,11,119,5,4,3,39,39,375,1,0,1,0,20,0,1,29,1,0,2,2,27,27,2,1,0,0,0,1,1,1,0,0,0,0,0,31,1,0,0,1,0,1,0,114,0,2,0,1,1,0,0,0,0,4,6,1,83,28,0,1,0},{2012,1242,12,792,1145,1588,742,676,762,649,93,467,537,1088,541,577,570,403,434,160,131,273,44,8458,175,157,110,18,98,158,147,0,96,61,122,7,121,72,100,10,24,9,2,5,38,65,131,1,2,1,1,30,2,1,37,1,1,0,0,37,40,2,1,1,2,0,1,1,1,0,0,0,1,0,29,1,1,0,1,0,2,0,121,1,1,0,2,2,0,1,0,0,4,6,1,82,28,0,1,0},{1899,1542,17,735,784,1654,801,811,978,387,84,338,607,1276,561,521,577,438,169,134,105,279,44,5406,195,152,117,17,93,63,52,0,40,56,105,8,31,75,98,12,18,4,3,5,37,33,24,1,0,0,0,10,0,1,116,1,0,2,2,27,30,2,1,0,0,0,1,1,1,0,0,0,0,0,7,1,0,0,1,0,1,0,134,0,1,0,1,1,0,0,0,0,4,6,1,80,28,0,1,0},{2405,1179,13023,858,423,920,706,710,630,418,7977,420,522,794,550,506,541,377,386,126,95,263,94,3137,164,180,408,1903,340,146,134,1,108,42,61,405,157,64,92,905,11,306,5,2,37,34,105,1,1,1,0,0,1,1,63,1,2,3,3,24,23,2,1,1,1,0,1,1,1,0,0,0,1,0,18,1,1,0,1,0,1,0,117,1,2,0,2,2,0,1,0,0,4,7,1,79,28,0,1,0} };

/\*a.push\_back({});

for (int i = 0; i < a[0].size(); i++) {

a[a.size() - 1].push\_back(0);

}\*/

/\*for (auto& vec : a) {

int i = 9;

while(i--)

vec.pop\_back();

}\*/

b = { {2.183,0.696,0.032,0.548,0.071,0.074,1.526,0.113},{1.497,0.344,0.101,0.515,0.006,0.085,1.711,0.109},{1.473,0.677,0.021,0.414,0.026,4.948,1.4,0.106},{1.201,0.394,0.124,0.829,0.058,2.883,2.03,0.639},{1.303,1.039,0.043,0.573,0.084,5.029,2.606,0.699},{1.354,0.857,0.085,0.576,0.114,0.425,2.011,0.607},{1.532,0.798,0.015,0.521,0.171,0.223,1.903,1.204},{2.31,0.491,0.071,0.855,1.933,0.138,1.983,1.233},{2.72,1.036,0.024,0.657,1.9,0.099,2.144,0.728},{2.553,0.704,0.11,1.134,2.357,0.06,2.077,0.973},{2.299,0.569,0.017,0.897,2.642,0.051,1.933,0.686},{2.124,0.79,0.074,0.453,0.575,0.041,3.283,0.902},{2.158,0.563,0.329,0.421,2.189,0.036,3.026,0.81},{2.296,0.488,0.079,0.423,2.377,0.036,2.277,0.709},{2.314,1.158,0.033,0.427,2.178,0.035,1.922,3.753},{2.133,0.742,0.119,0.677,2.313,0.034,1.919,5.065},{1.939,0.622,0.029,0.568,1.993,0.031,1.432,0.756},{1.965,0.857,0.072,0.431,1.406,0.035,1.516,0.417},{2.268,0.59,0.015,0.763,0.943,0.032,1.409,0.432},{1.981,0.494,0.087,0.523,0.55,0.032,1.476,0.45},{1.911,0.603,0.062,0.515,0.557,0.031,1.644,0.643},{2.014,0.604,0.132,0.46,0.585,0.029,1.603,0.494},{1.999,0.564,0.029,0.748,0.681,0.029,1.692,0.705},{2.109,0.847,0.046,0.573,0.61,0.049,1.747,0.966} };

/\*b.push\_back({});

for (int i = 0; i < b[0].size(); i++) {

b[b.size() - 1].push\_back(0);

}\*/

a\_ = a;

allocat\_ans = vector<vector<int>>(a.size(),vector<int>(a[0].size(),0));

/\*for (int i = 0; i < a.size(); i++) {

for (int j = 0; j < a[i].size(); j++) {

a[i][j] \*= 1.1;

}

}\*/

cumsum = vector<double>(weeks, 0);

double expect = 0;

double lan = 0;

for (int j = 0; j < a[0].size(); j++)cumsum[0] += a[0][j]/actual\_consume[kind\_dict[j]-1];

for (int i = 1; i < a.size(); i++) {

double sum = 0;

for (int j = 0; j < a[i].size(); j++) {

sum += a[i][j]/actual\_consume[kind\_dict[j]-1];

}

cumsum[i] = cumsum[i - 1] + sum;

}

for (int i = weeks-1; i >= 0; i--) {

double sum = 0;

for (int j = 0; j < a[i].size(); j++) {

sum += a[i][j] / actual\_consume[kind\_dict[j] - 1];

}

//cout << lan << endl;

if (sum-capacity\_need > 0) {

if (lan < 0) {

if (sum - capacity\_need + lan > 0) {

expect += sum - capacity\_need + lan;

lan = 0;

}

else {

lan += sum - capacity\_need;

}

}

else {

expect += sum - capacity\_need;;

}

}

else {

lan += sum - capacity\_need;

}

}

cout << "不计损耗、计时序拓扑截尾后的产能上限为" << capacity\_need+lan/24<<endl;

for (auto& c : cumsum) {

if (c / capacity\_need >= useful\_bound.size() + 1)useful\_bound.push\_back(true);

else useful\_bound.push\_back(false);

//cout << c / capacity\_need <<" "<<useful\_bound.size()<<endl;

}

}

vector<pair<int, pair<int, double>>> optical\_trans\_order(int weeks,int nums = 4) {

vector<pair<int, pair<int, double>>> opt\_set;//triple->(week,no.,rate)

for (int k = 0; k < weeks; k++) {

vector<pair<int, double>> tup;

for (int l = 0; l < b[k].size(); l++) {

tup.push\_back(make\_pair(l,b[k][l]));

}

sort(tup.begin(), tup.end(), [](auto& x, auto& y) {return x.second < y.second; });

//放入损耗率最小的前四个即可

for (int i = 0; i < nums; i++) {

opt\_set.push\_back(make\_pair(k, tup[i]));

}

}

sort(opt\_set.begin(), opt\_set.end(), [](auto& x, auto& y) {return x.second.second < y.second.second; });

return opt\_set;

}

bool execute() {

auto steps = optical\_trans\_order(weeks,4);

for (auto& step : steps) {

int week = step.first;//获取周数

int no = step.second.first;//获取仓库号

for (int kind = 1; kind < 4; kind++) {

for (int i = week; i >= 0; i--) {//倒序依次装配A B C货物

for (int pro = 0; pro < a[week].size(); pro++) {//其实可以优化一下，减少外部碎片，这里暂时先不管了

if (kind\_dict[pro] != kind||a[i][pro]==0)continue;

if (weekly\_actual\_capacity[week] >= capacity\_need-1)continue;

//检验边界情况是否越出

bool drop = false;

int count = 0;

double all = 0;

for (int aloc\_i = 0; aloc\_i < i; aloc\_i++) {

all += capacity\_need - weekly\_actual\_capacity[aloc\_i];

}

for (int aloc\_i = i; aloc\_i < week; aloc\_i++) {

all += capacity\_need - weekly\_actual\_capacity[aloc\_i];

if (useful\_bound[aloc\_i]) {//检测是否越界了

if (all > cumsum[aloc\_i]) {

drop = true;

break;

}

}

}

//if(week>=1)

/\*for (int aloc\_i = i; aloc\_i <week; aloc\_i++) {

all += capacity\_need - weekly\_actual\_capacity[aloc\_i];

if (!useful\_bound[aloc\_i])break;

if (all > cumsum[aloc\_i]&&useful\_bound[aloc\_i]) {

drop = true;

break;

//count++;

}

}\*/

//if (count == week - i-1||count == 0)drop = false;

//else drop = true;

//检测有没有库存超限，超限情况下应当另外讨论，这里为了简化模型选择直接放弃此次分配

int minist = 6000;

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

minist = min(block[aloc\_i][no][3], minist);

if (block[aloc\_i][no][3] <=0){//< a[i][pro]) {

drop = true;

break;

}

}

if (drop) break;

minist = min(minist, (int)((capacity\_need - weekly\_actual\_capacity[week])\*actual\_consume[kind-1]));

minist = max(minist, 0);

//必要的时候分割转运操作避免爆仓

if (minist > 0 && minist < a[i][pro]) {

//cout << a[i][pro]<<" ";

a[i][pro] -= minist;

int tmp = minist;

minist = a[i][pro];

a[i][pro] = tmp;

//cout << a[i][pro] << endl;

}

else minist = 0;

//cout << a[i][pro] << endl;

//通过检测则放入

weekly\_actual\_capacity[week] += (1. - 0.01\*b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1])+0.5;//(1-损耗率）\*总量/单位消耗

ave += (1. - 0.01 \* b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1]);

//cout << weekly\_actual\_capacity[week] << endl;

//在[i,week]期间，将货物存放到no号仓库内，更新时空图和仓储分区与实际产能,校验累积和边界

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

cumsum[aloc\_i] -= a[i][pro] / (actual\_consume[kind - 1]);

block[aloc\_i][no][kind-1] += a[i][pro];

block[aloc\_i][no][3] -= a[i][pro];

allocate\_status\_map[aloc\_i][no][kind - 1].push(make\_pair(make\_pair(i, pro),a[i][pro]));

//cout << i << " " << pro << " " << a\_[i][pro] << endl;

}

allocat\_ans[i][pro] += a[i][pro];

//清空供货商相关存量

a[i][pro] = minist;

}

}

}

}//with\_bound

for (int i = 0; i < useful\_bound.size(); i++) {

if (weekly\_actual\_capacity[i] >= capacity\_need - 1) {

useful\_bound[i] = true;

}

else {

useful\_bound[i] = false;

//cout << i << endl;

}

}

capacity\_need \*= 2;//获取超出的材料

steps = optical\_trans\_order(weeks, 8);

//sort(steps.begin(), steps.end(), [](auto& x, auto& y) {return (x.first < y.first); });

for (auto& step : steps) {

int week = step.first;//获取周数

int no = step.second.first;//获取仓库号

for (int i = week; i >= 0; i--) {

for (int kind = 1; kind < 4; kind++) {//倒序依次装配A B C货物

for (int pro = 0; pro < a[week].size(); pro++) {//其实可以优化一下，减少外部碎片，这里暂时先不管了

if (kind\_dict[pro] != kind || a[i][pro] == 0)continue;

if (useful\_bound[week])continue;

bool drop = false;

//检测是否有必要扩

//检测有没有库存超限，超限情况下应当另外讨论，这里为了简化模型选择直接放弃此次分配

int minist = 6000;

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

minist = min(block[aloc\_i][no][3], minist);

if (block[aloc\_i][no][3] <= 0) {//< a[i][pro]) {

drop = true;

break;

}

}

//cout << minist << endl;

if (drop) break;

minist = min(minist, (int)((capacity\_need - weekly\_actual\_capacity[week]) \* actual\_consume[kind - 1]));

minist = max(minist, 0);

//必要的时候分割转运操作避免爆仓

if (minist > 0 && minist < a[i][pro]) {

//cout << a[i][pro]<<" ";

a[i][pro] -= minist;

int tmp = minist;

minist = a[i][pro];

a[i][pro] = tmp;

//cout << a[i][pro] << endl;

}

else minist = 0;

//cout << a[i][pro] << endl;

//通过检测则放入

weekly\_actual\_capacity[week] += (1. - 0.01 \* b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1])+0.5;//(1-损耗率）\*总量/单位消耗

ave += (1. - 0.01 \* b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1]);

if (weekly\_actual\_capacity[week] >= capacity\_need / 2 - 1)useful\_bound[week] = true;

//在[i,week]期间，将货物存放到no号仓库内，更新时空图和仓储分区与实际产能,校验累积和边界

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

cumsum[aloc\_i] -= a[i][pro] / (actual\_consume[kind - 1]);

block[aloc\_i][no][kind - 1] += a[i][pro];

block[aloc\_i][no][3] -= a[i][pro];

allocate\_status\_map[aloc\_i][no][kind - 1].push(make\_pair(make\_pair(i, pro), a[i][pro]));

//cout << i << " " << pro << " " << a\_[i][pro] << endl;

}

allocat\_ans[i][pro] += a[i][pro];

//清空供货商相关存量

a[i][pro] = minist;

}

}

}

}//without bound

cout << "周产能:";

int sum = 0;

for (auto& e : weekly\_actual\_capacity) {

cout << e << " ";

sum += e;

}

cout << endl;

cout << ave/24 << endl;

cout << "转运决策如下：";

sort(steps.begin(), steps.end(), [](auto& x, auto& y) {return x.first < y.first; });

//cout << "week1:" << endl;

int nums = 8;

for (int i = 0; i < steps.size(); i++) {

if (i % nums == 0)cout <<"第"<<steps[i].first<<"周"<< endl;

if (block[i/nums][steps[i].second.first][3] == 6000)continue;

cout <<"仓库" << steps[i].second.first <<" 损耗率"<< steps[i].second.second<<" 运送"<< endl;

cout << "仓储分区情况：" << endl;

cout << "A:" <<block[i/nums][steps[i].second.first][0]<< " B:"<< block[i / nums][steps[i].second.first][1] << " C:" << block[i / nums][steps[i].second.first][2] << " empty:" << block[i / nums][steps[i].second.first][3] << endl;

cout << "具体策略如下：" << endl;

int e = steps[i].second.first;

for (int j = 0; j < 3; j++) {

cout << "材料种类" << j << endl;

while (!allocate\_status\_map[i / nums][e][j].empty()) {

auto& tmp = allocate\_status\_map[i / nums][e][j].top();

allocate\_status\_map[i / nums][e][j].pop();

cout << "存入周:" << tmp.first.first << "\t商家:" << tmp.first.second << "\t 数量" << tmp.second <<'\t'<< endl;

//allocat\_ans[tmp.first.first][tmp.first.second] += tmp.second;

}

}

}

cout << "订货情况：（每行 为一个商家的数据）"<<endl;

for (int i = 0; i < a[0].size(); i++) {

for (int j = 0; j < a.size(); j++) {

cout << allocat\_ans[j][i]<< '\t';

}

cout << endl;

}

cout << "材料空闲情况" << endl;

for (int i = 0; i < a[0].size(); i++) {

for (int j = 0; j < a.size(); j++) {

cout << (-allocat\_ans[j][i] + a\_[j][i])<< '\t';

}

cout << endl;

}

return true;

}

};

int main(void) {

Solution s = Solution();

s.execute();

char ch;

cin >> ch;

return 0;

}

以下为第四问的C++代码部分：

#include<iostream>

#include<vector>

#include<queue>

#include<map>

using namespace std;

class Solution {

private:

vector<double> d;//表示1m^3货物的制造产能

vector<double> e;//表示订购单位j种货物的价格

vector<double> m;//表示k周总供货量

vector<double> z;//表示k周总损耗

vector<double> w;//表示k周总成本

vector<double> u\_1;//表示k周采购成本

vector<double> u\_2;//表示k周运输成本

vector<double> u\_3;//表示k周存储成本

double f0, e0, g0;//运输、存储、损耗

//仓库货物时空分布图

int weeks;

double capacity\_need;

vector<vector<int>> a;//表示在第k周第i家供货商的供货量

vector<vector<int>> a\_;

vector<int> kind\_dict;//供货商商品种类映射 A-1,B-2,C-3

vector<double> actual\_consume;//A-0.6,B-0.66,C-0.72

vector<vector<vector<priority\_queue<pair<pair<int, int>,int>, vector<pair<pair<int, int>,int>>>>>> allocate\_status\_map;//[周][仓库]->(（周数，商家）,数量)

vector<vector<vector<int>>> block;//[周][仓库号]仓储分区(A-B-C-EMPTY)

vector<int> weekly\_actual\_capacity;//实际产能--按周记录

vector<vector<double>> b;//表示在k周第L家转运商的转运损耗率

vector<double> cumsum;//周累积和边界检测，需要在保证每周产能的前提下减少购买成本和贮存成本

vector<bool> useful\_bound;

vector<vector<int>> allocat\_ans;

double ave;

public:

Solution() {

ave = 0;

weeks = 24;

capacity\_need = 28200.\*2;

//kind\_dict = { 3,3,3,3,2,3,3,2,1,1,1,2,3,1,2,2,1,2,3,1,1,1,1,3,1,1,3,2,3,3,1,3,1,1,3,1,2,1,3,1,2,2,2,3,2,3,1,1,1,2 };//2

kind\_dict = { 3,3,3,3,2,3,3,2,1,1,1,2,3,1,2,2,1,2,3,1,1,1,1,3,1,1,3,2,3,3,1,3,1,1,3,1,2,1,3,1,2,2,2,3,2,3,1,1,1,2,

1,1,2,1,1,1,1,2,2,1,1,3,1,1,2,3,2,2,2,1,3,3,1,1,2,1,3,1,3,2,1,1,3,3,2,2,1,1,3,2,3,3,3,1,1,1,3,3,2,3};

actual\_consume = { 0.6,0.66,0.72 };

block = vector<vector<vector<int>>>(weeks, vector<vector<int>>(50, vector<int>(4, 0)));

for (int i = 0; i < block.size(); i++) {

for (int j = 0; j < block[i].size(); j++) {

block[i][j][3] = 6000;

}

}

weekly\_actual\_capacity = vector<int>(weeks, 0);

allocate\_status\_map = vector<vector<vector<priority\_queue<pair<pair<int, int>, int>, vector<pair<pair<int, int>, int>>>>>>(weeks, vector<vector<priority\_queue<pair<pair<int, int>, int>, vector<pair<pair<int, int>, int>>>>>(8, vector<priority\_queue<pair<pair<int, int>, int>, vector<pair<pair<int, int>, int>>>>(3)));

//a = { {1475,1276,10,6245,265,533,666,677,748,426,53,386,662,581,567,373,491,368,206,62,158,226,48,1185,139,352,109,12,81,37,63,0,36,1008,30,3,26,64,103,7,62,1,4,15,11,38,44,0,0,0},{1438,1006,12,833,219,569,779,808,853,261,6931,526,674,483,613,428,544,423,401,65,196,246,50,747,175,293,125,12,110,155,189,0,115,53,26,7,149,74,102,1221,19,5,1,23,11,70,74,2,2,1},{1376,1621,11,749,201,527,753,766,995,205,78,373,679,426,637,385,548,432,273,71,193,273,476,518,201,309,347,1682,486,37,51,0,45,56,19,296,32,83,148,7,27,71,5,12,12,35,127,1,0,1},{1459,1501,13421,862,227,618,861,817,1075,313,82,472,810,420,646,457,606,476,477,77,159,282,279,402,213,198,112,12,229,186,145,0,141,60,20,8,168,87,112,7,31,417,2,1,13,36,146,0,2,1},{1612,936,24,889,329,561,877,909,1118,6951,98,445,922,387,666,400,677,510,115,82,188,323,8781,344,217,178,143,24,104,56,52,0,46,69,28,11,36,100,111,19,60,10,7,1,14,32,739,0,0,2},{1573,1148,17,813,468,565,736,729,927,486,96,414,761,392,527,455,562,409,560,91,259,268,85,320,184,145,115,14,103,150,124,0,112,58,73,9,139,84,95,7,348,9,6,14,16,43,392,1,1,1},{1874,1144,18,740,605,511,765,789,890,445,94,491,585,691,525,420,521,408,120,244,573,256,59,317,166,143,115,20,83,48,44,0,38,59,55,11,29,67,95,14,1652,8,4,5,18,25,137,5,0,1},{2068,1224,24,854,697,609,801,834,1004,307,104,507,786,708,554,660,571,436,408,463,280,268,49,315,192,157,120,22,83,141,154,0,105,47,76,18,135,78,101,15,1690,16,3,6,21,44,25,1,0,2},{1951,1454,14,747,757,513,706,770,963,426,87,385,697,727,555,664,578,451,201,1839,200,276,48,316,185,158,119,17,98,52,50,359,40,51,79,8,29,76,97,11,1472,5,4,5,24,34,110,2,1,2},{1923,1261,17,818,727,628,691,695,876,314,92,337,690,842,499,672,544,391,413,481,147,242,46,298,186,135,103,14,86,127,177,587,94,45,68,9,116,77,85,11,1196,8,5,4,28,36,83,2,0,1},{2098,1519,17,727,787,527,751,699,815,372,86,472,738,804,497,562,538,408,168,396,128,254,46,288,164,150,118,15,90,44,65,431,37,47,77,8,29,79,89,12,1969,6,59,5,32,23,35,0,0,1},{2783,991,17,825,937,679,724,789,898,294,94,286,720,1111,521,589,552,395,457,202,145,255,45,288,171,176,121,17,80,125,155,73,89,52,66,11,117,85,89,11,2036,10,5,4,35,37,34,1,1,2},{1824,1252,19,798,949,571,819,804,989,306,89,385,777,969,557,563,609,436,107,264,95,277,45,299,182,159,112,17,98,56,53,852,47,56,86,10,28,87,98,12,1622,5,4,5,38,15,81,0,0,1},{1765,1399,23,820,931,669,773,826,996,262,97,396,646,1066,554,646,637,448,872,438,174,284,45,302,174,162,129,16,99,163,175,626,101,52,93,12,136,75,95,11,9287,8,2,6,40,23,46,1,2,2},{1771,1732,21,698,931,595,690,821,891,254,86,1314,666,1021,531,553,634,404,177,498,217,267,45,290,181,145,116,17,83,62,51,643,37,52,102,8,33,76,95,9,82,5,6,5,42,12,82,0,0,1},{1723,1100,19,842,954,717,712,839,896,360,95,537,630,868,525,576,615,399,398,353,215,241,45,285,174,160,128,17,92,158,168,462,86,50,104,14,125,73,91,13,88,7,3,6,42,23,41,1,1,2},{1781,1583,13,741,1060,574,751,808,868,329,86,482,746,1004,517,502,597,439,256,1235,208,262,44,274,181,149,103,15,92,50,47,805,40,55,90,8,30,69,92,14,94,5,2,6,42,34,71,0,1,0},{2315,1472,15,706,1231,555,690,822,890,373,94,387,694,934,526,571,554,415,451,3237,144,249,45,268,150,146,113,17,85,151,189,408,96,45,100,9,127,87,101,12,610,7,2,3,42,23,121,2,3,1},{2152,1410,13,677,1430,492,654,860,815,297,85,431,826,1089,518,575,520,428,97,1262,63,292,44,267,187,145,103,14,89,57,67,11,38,46,114,7,30,89,92,10,148,5,2,5,41,18,139,1,0,0},{2577,1406,16,763,1149,569,640,603,646,307,90,324,468,1310,430,543,478,259,549,177,96,200,44,33948,131,114,99,12,85,152,147,25,83,40,134,11,125,78,82,10,113,8,3,4,40,24,708,1,2,1},{2244,1615,15,673,1359,540,625,681,728,373,87,346,543,1182,513,490,561,400,111,187,118,255,44,18858,179,142,101,18,86,50,51,11,37,45,111,6,28,68,101,11,119,5,4,3,39,39,375,1,0,1},{2012,1242,12,792,1145,1588,742,676,762,649,93,467,537,1088,541,577,570,403,434,160,131,273,44,8458,175,157,110,18,98,158,147,0,96,61,122,7,121,72,100,10,-24,9,2,5,38,65,131,1,2,1},{1899,1542,17,735,784,1654,801,811,978,387,84,338,607,1276,561,521,577,438,169,134,105,279,44,5406,195,152,117,17,93,63,52,0,40,56,105,8,31,75,98,12,-18,4,3,5,37,33,24,1,0,0},{2405,1179,13023,858,423,920,706,710,630,418,7977,420,522,794,550,506,541,377,386,126,95,263,94,3137,164,180,408,1903,340,146,134,-1,108,42,61,405,157,64,92,905,-11,306,5,2,37,34,105,1,1,1} };

a = { {1475,1276,10,6245,265,533,666,677,748,426,53,386,662,581,567,373,491,368,206,62,158,226,48,1185,139,352,109,12,81,37,63,0,36,1008,30,3,26,64,103,7,62,1,4,15,11,38,44,0,0,0,0,1,0,0,64,0,1,0,0,26,24,1,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,128,0,1,0,1,1,0,0,0,0,4,1,0,81,25,0,1,0},{1438,1006,12,833,219,569,779,808,853,261,6931,526,674,483,613,428,544,423,401,65,196,246,50,747,175,293,125,12,110,155,189,0,115,53,26,7,149,74,102,1221,19,5,1,23,11,70,74,2,2,1,2,0,2,1,86,3,2,3,3,26,21,2,2,1,1,1,2,2,1,1,1,1,1,0,35,1,1,1,1,1,1,0,133,1,1,1,4,2,0,1,1,0,4,1,2,78,27,9,1,2},{1376,1621,11,749,201,527,753,766,995,205,78,373,679,426,637,385,548,432,273,71,193,273,476,518,201,309,347,1682,486,37,51,0,45,56,19,296,32,83,148,7,27,71,5,12,12,35,127,1,0,1,0,4,1,1,19,0,0,1,1,31,26,1,1,1,0,0,1,1,1,0,0,0,0,0,41,1,0,0,1,1,1,0,132,0,1,0,1,1,0,1,0,0,4,1,0,76,27,0,1,0},{1459,1501,13421,862,227,618,861,817,1075,313,82,472,810,420,646,457,606,476,477,77,159,282,279,402,213,198,112,12,229,186,145,0,141,60,20,8,168,87,112,7,31,417,2,1,13,36,146,0,2,1,3,1,2,0,24,2,1,2,2,32,29,2,2,2,3,1,1,1,1,1,1,1,1,0,48,1,1,1,1,0,3,0,132,1,2,0,4,2,0,1,0,0,4,1,1,75,29,0,1,0},{1612,936,24,889,329,561,877,909,1118,6951,98,445,922,387,666,400,677,510,115,82,188,323,8781,344,217,178,143,24,104,56,52,0,46,69,28,11,36,100,111,19,60,10,7,1,14,32,739,0,0,2,1,1,1,2,20,1,0,1,1,32,33,1,1,1,0,0,1,1,1,0,0,0,1,0,35,1,0,0,1,0,1,0,132,0,1,1,1,1,0,1,0,0,4,2,1,75,31,0,1,0},{1573,1148,17,813,468,565,736,729,927,486,96,414,761,392,527,455,562,409,560,91,259,268,85,320,184,145,115,14,103,150,124,0,112,58,73,9,139,84,95,7,348,9,6,14,16,43,392,1,1,1,0,2,1,0,16,1,1,1,1,30,28,2,1,1,2,1,1,1,1,2,0,1,2,0,60,2,0,0,1,1,2,0,132,1,1,15,2,2,0,1,1,0,2,2,0,75,28,0,1,0},{1874,1144,18,740,605,511,765,789,890,445,94,491,585,691,525,420,521,408,120,244,573,256,59,317,166,143,115,20,83,48,44,0,38,59,55,11,29,67,95,14,1652,8,4,5,18,25,137,5,0,1,1,3,1,1,14,1,0,2,2,27,24,2,1,1,0,0,1,1,1,0,0,0,0,0,38,1,0,0,1,0,1,0,132,0,2,29,1,1,0,1,0,0,4,2,1,76,27,0,1,0},{2068,1224,24,854,697,609,801,834,1004,307,104,507,786,708,554,660,571,436,408,463,280,268,49,315,192,157,120,22,83,141,154,0,105,47,76,18,135,78,101,15,1690,16,3,6,21,44,25,1,0,2,1,1,3,2,20,2,2,3,3,27,39,2,1,2,1,1,1,2,1,2,1,1,1,0,33,1,1,1,1,1,2,0,132,1,1,26,2,1,0,1,0,0,4,2,2,79,28,0,1,0},{1951,1454,14,747,757,513,706,770,963,426,87,385,697,727,555,664,578,451,201,1839,200,276,48,316,185,158,119,17,98,52,50,359,40,51,79,8,29,76,97,11,1472,5,4,5,24,34,110,2,1,2,1,2,1,2,16,0,1,1,1,22,29,2,1,1,0,0,1,1,1,0,0,0,0,0,66,1,0,0,1,0,1,0,132,0,1,15,1,1,0,1,1,0,4,3,1,81,28,0,1,0},{1923,1261,17,818,727,628,691,695,876,314,92,337,690,842,499,672,544,391,413,481,147,242,46,298,186,135,103,14,86,127,177,587,94,45,68,9,116,77,85,11,1196,8,5,4,28,36,83,2,0,1,1,1,1,2,21,1,1,0,0,25,23,2,1,1,1,1,1,1,1,1,0,0,1,0,36,1,1,0,1,1,1,0,131,1,1,1,3,1,0,1,1,0,4,3,2,82,28,0,1,0},{2098,1519,17,727,787,527,751,699,815,372,86,472,738,804,497,562,538,408,168,396,128,254,46,288,164,150,118,15,90,44,65,431,37,47,77,8,29,79,89,12,1969,6,59,5,32,23,35,0,0,1,1,0,1,0,17,0,1,1,1,32,28,2,1,1,0,0,1,1,1,0,0,0,0,0,35,1,0,0,1,0,1,0,126,0,1,0,1,1,0,1,1,0,4,3,1,81,28,0,1,0},{2783,991,17,825,937,679,724,789,898,294,94,286,720,1111,521,589,552,395,457,202,145,255,45,288,171,176,121,17,80,125,155,73,89,52,66,11,117,85,89,11,2036,10,5,4,35,37,34,1,1,2,1,1,1,2,19,1,0,1,1,25,25,2,2,1,1,0,1,1,1,1,1,0,2,0,20,1,1,0,1,1,2,0,130,1,2,0,2,1,0,1,0,0,2,3,2,80,27,0,1,0},{1824,1252,19,798,949,571,819,804,989,306,89,385,777,969,557,563,609,436,107,264,95,277,45,299,182,159,112,17,98,56,53,852,47,56,86,10,28,87,98,12,1622,5,4,5,38,15,81,0,0,1,0,2,0,1,18,1,1,1,1,24,25,2,1,1,0,0,1,1,1,0,0,0,0,0,35,1,0,0,1,0,1,0,134,0,1,5,1,1,0,0,1,0,4,4,1,82,30,0,1,0},{1765,1399,23,820,931,669,773,826,996,262,97,396,646,1066,554,646,637,448,872,438,174,284,45,302,174,162,129,16,99,163,175,626,101,52,93,12,136,75,95,11,9287,8,2,6,40,23,46,1,2,2,1,1,2,1,13,0,3,2,2,34,25,2,2,2,2,1,1,1,1,1,1,0,2,0,34,2,1,0,1,1,2,0,98,1,2,0,1,2,0,1,0,0,4,4,2,82,29,0,1,0},{1771,1732,21,698,931,595,690,821,891,254,86,1314,666,1021,531,553,634,404,177,498,217,267,45,290,181,145,116,17,83,62,51,643,37,52,102,8,33,76,95,9,82,5,6,5,42,12,82,0,0,1,0,3,0,0,17,1,1,2,2,30,29,2,1,1,0,0,1,1,1,0,0,0,0,0,27,1,0,0,1,0,1,0,121,0,2,0,1,1,0,0,1,0,4,4,1,83,28,0,1,0},{1723,1100,19,842,954,717,712,839,896,360,95,537,630,868,525,576,615,399,398,353,215,241,45,285,174,160,128,17,92,158,168,462,86,50,104,14,125,73,91,13,88,7,3,6,42,23,41,1,1,2,1,1,2,0,18,1,2,2,2,44,28,2,2,1,0,1,1,1,1,1,0,0,1,0,33,2,1,0,1,0,3,0,135,1,2,0,3,2,0,1,1,0,4,5,1,83,28,0,1,0},{1781,1583,13,741,1060,574,751,808,868,329,86,482,746,1004,517,502,597,439,256,1235,208,262,44,274,181,149,103,15,92,50,47,805,40,55,90,8,30,69,92,14,94,5,2,6,42,34,71,0,1,0,0,3,1,1,16,0,1,1,1,24,25,2,1,1,0,0,1,1,1,0,0,0,0,0,33,1,0,0,1,0,1,0,131,0,2,0,1,1,0,0,0,0,4,5,1,83,28,0,1,0},{2315,1472,15,706,1231,555,690,822,890,373,94,387,694,934,526,571,554,415,451,3237,144,249,45,268,150,146,113,17,85,151,189,408,96,45,100,9,127,87,101,12,610,7,2,3,42,23,121,2,3,1,1,2,1,1,18,3,0,3,3,28,26,2,1,1,0,0,1,1,1,0,0,0,1,0,10,2,1,0,1,1,2,0,77,0,2,0,2,2,0,1,0,0,2,5,2,82,28,0,1,0},{2152,1410,13,677,1430,492,654,860,815,297,85,431,826,1089,518,575,520,428,97,1262,63,292,44,267,187,145,103,14,89,57,67,11,38,46,114,7,30,89,92,10,148,5,2,5,41,18,139,1,0,0,0,4,0,0,19,0,1,1,1,24,24,2,1,1,0,0,1,1,1,0,0,0,0,0,30,1,0,0,1,0,1,0,71,0,2,0,1,1,0,0,0,0,4,5,1,83,27,0,1,0},{2577,1406,16,763,1149,569,640,603,646,307,90,324,468,1310,430,543,478,259,549,177,96,200,44,33948,131,114,99,12,85,152,147,25,83,40,134,11,125,78,82,10,113,8,3,4,40,24,708,1,2,1,2,6,1,1,21,2,1,3,3,22,26,2,1,1,1,0,1,1,1,0,0,0,2,0,5,2,0,0,1,1,2,0,125,1,1,0,3,2,0,1,1,0,4,5,2,83,29,0,1,0},{2244,1615,15,673,1359,540,625,681,728,373,87,346,543,1182,513,490,561,400,111,187,118,255,44,18858,179,142,101,18,86,50,51,11,37,45,111,6,28,68,101,11,119,5,4,3,39,39,375,1,0,1,0,20,0,1,29,1,0,2,2,27,27,2,1,0,0,0,1,1,1,0,0,0,0,0,31,1,0,0,1,0,1,0,114,0,2,0,1,1,0,0,0,0,4,6,1,83,28,0,1,0},{2012,1242,12,792,1145,1588,742,676,762,649,93,467,537,1088,541,577,570,403,434,160,131,273,44,8458,175,157,110,18,98,158,147,0,96,61,122,7,121,72,100,10,24,9,2,5,38,65,131,1,2,1,1,30,2,1,37,1,1,0,0,37,40,2,1,1,2,0,1,1,1,0,0,0,1,0,29,1,1,0,1,0,2,0,121,1,1,0,2,2,0,1,0,0,4,6,1,82,28,0,1,0},{1899,1542,17,735,784,1654,801,811,978,387,84,338,607,1276,561,521,577,438,169,134,105,279,44,5406,195,152,117,17,93,63,52,0,40,56,105,8,31,75,98,12,18,4,3,5,37,33,24,1,0,0,0,10,0,1,116,1,0,2,2,27,30,2,1,0,0,0,1,1,1,0,0,0,0,0,7,1,0,0,1,0,1,0,134,0,1,0,1,1,0,0,0,0,4,6,1,80,28,0,1,0},{2405,1179,13023,858,423,920,706,710,630,418,7977,420,522,794,550,506,541,377,386,126,95,263,94,3137,164,180,408,1903,340,146,134,1,108,42,61,405,157,64,92,905,11,306,5,2,37,34,105,1,1,1,0,0,1,1,63,1,2,3,3,24,23,2,1,1,1,0,1,1,1,0,0,0,1,0,18,1,1,0,1,0,1,0,117,1,2,0,2,2,0,1,0,0,4,7,1,79,28,0,1,0} };

/\*a.push\_back({});

for (int i = 0; i < a[0].size(); i++) {

a[a.size() - 1].push\_back(0);

}\*/

/\*for (auto& vec : a) {

int i = 9;

while(i--)

vec.pop\_back();

}\*/

b = { {2.183,0.696,0.032,0.548,0.071,0.074,1.526,0.113},{1.497,0.344,0.101,0.515,0.006,0.085,1.711,0.109},{1.473,0.677,0.021,0.414,0.026,4.948,1.4,0.106},{1.201,0.394,0.124,0.829,0.058,2.883,2.03,0.639},{1.303,1.039,0.043,0.573,0.084,5.029,2.606,0.699},{1.354,0.857,0.085,0.576,0.114,0.425,2.011,0.607},{1.532,0.798,0.015,0.521,0.171,0.223,1.903,1.204},{2.31,0.491,0.071,0.855,1.933,0.138,1.983,1.233},{2.72,1.036,0.024,0.657,1.9,0.099,2.144,0.728},{2.553,0.704,0.11,1.134,2.357,0.06,2.077,0.973},{2.299,0.569,0.017,0.897,2.642,0.051,1.933,0.686},{2.124,0.79,0.074,0.453,0.575,0.041,3.283,0.902},{2.158,0.563,0.329,0.421,2.189,0.036,3.026,0.81},{2.296,0.488,0.079,0.423,2.377,0.036,2.277,0.709},{2.314,1.158,0.033,0.427,2.178,0.035,1.922,3.753},{2.133,0.742,0.119,0.677,2.313,0.034,1.919,5.065},{1.939,0.622,0.029,0.568,1.993,0.031,1.432,0.756},{1.965,0.857,0.072,0.431,1.406,0.035,1.516,0.417},{2.268,0.59,0.015,0.763,0.943,0.032,1.409,0.432},{1.981,0.494,0.087,0.523,0.55,0.032,1.476,0.45},{1.911,0.603,0.062,0.515,0.557,0.031,1.644,0.643},{2.014,0.604,0.132,0.46,0.585,0.029,1.603,0.494},{1.999,0.564,0.029,0.748,0.681,0.029,1.692,0.705},{2.109,0.847,0.046,0.573,0.61,0.049,1.747,0.966} };

/\*b.push\_back({});

for (int i = 0; i < b[0].size(); i++) {

b[b.size() - 1].push\_back(0);

}\*/

a\_ = a;

allocat\_ans = vector<vector<int>>(a.size(),vector<int>(a[0].size(),0));

/\*for (int i = 0; i < a.size(); i++) {

for (int j = 0; j < a[i].size(); j++) {

a[i][j] \*= 1.1;

}

}\*/

cumsum = vector<double>(weeks, 0);

double expect = 0;

double lan = 0;

for (int j = 0; j < a[0].size(); j++)cumsum[0] += a[0][j]/actual\_consume[kind\_dict[j]-1];

for (int i = 1; i < a.size(); i++) {

double sum = 0;

for (int j = 0; j < a[i].size(); j++) {

sum += a[i][j]/actual\_consume[kind\_dict[j]-1];

}

cumsum[i] = cumsum[i - 1] + sum;

}

for (int i = weeks-1; i >= 0; i--) {

double sum = 0;

for (int j = 0; j < a[i].size(); j++) {

sum += a[i][j] / actual\_consume[kind\_dict[j] - 1];

}

//cout << lan << endl;

if (sum-capacity\_need > 0) {

if (lan < 0) {

if (sum - capacity\_need + lan > 0) {

expect += sum - capacity\_need + lan;

lan = 0;

}

else {

lan += sum - capacity\_need;

}

}

else {

expect += sum - capacity\_need;;

}

}

else {

lan += sum - capacity\_need;

}

}

cout << "不计损耗、计时序拓扑截尾后的产能上限为" << capacity\_need+lan/24<<endl;

for (auto& c : cumsum) {

if (c / capacity\_need >= useful\_bound.size() + 1)useful\_bound.push\_back(true);

else useful\_bound.push\_back(false);

//cout << c / capacity\_need <<" "<<useful\_bound.size()<<endl;

}

}

vector<pair<int, pair<int, double>>> optical\_trans\_order(int weeks,int nums = 4) {

vector<pair<int, pair<int, double>>> opt\_set;//triple->(week,no.,rate)

for (int k = 0; k < weeks; k++) {

vector<pair<int, double>> tup;

for (int l = 0; l < b[k].size(); l++) {

tup.push\_back(make\_pair(l,b[k][l]));

}

sort(tup.begin(), tup.end(), [](auto& x, auto& y) {return x.second < y.second; });

//放入损耗率最小的前四个即可

for (int i = 0; i < nums; i++) {

opt\_set.push\_back(make\_pair(k, tup[i]));

}

}

sort(opt\_set.begin(), opt\_set.end(), [](auto& x, auto& y) {return x.second.second < y.second.second; });

return opt\_set;

}

bool execute() {

auto steps = optical\_trans\_order(weeks,4);

for (auto& step : steps) {

int week = step.first;//获取周数

int no = step.second.first;//获取仓库号

for (int kind = 1; kind < 4; kind++) {

for (int i = week; i >= 0; i--) {//倒序依次装配A B C货物

for (int pro = 0; pro < a[week].size(); pro++) {//其实可以优化一下，减少外部碎片，这里暂时先不管了

if (kind\_dict[pro] != kind||a[i][pro]==0)continue;

if (weekly\_actual\_capacity[week] >= capacity\_need-1)continue;

//检验边界情况是否越出

bool drop = false;

int count = 0;

double all = 0;

for (int aloc\_i = 0; aloc\_i < i; aloc\_i++) {

all += capacity\_need - weekly\_actual\_capacity[aloc\_i];

}

for (int aloc\_i = i; aloc\_i < week; aloc\_i++) {

all += capacity\_need - weekly\_actual\_capacity[aloc\_i];

if (useful\_bound[aloc\_i]) {//检测是否越界了

if (all > cumsum[aloc\_i]) {

drop = true;

break;

}

}

}

//if(week>=1)

/\*for (int aloc\_i = i; aloc\_i <week; aloc\_i++) {

all += capacity\_need - weekly\_actual\_capacity[aloc\_i];

if (!useful\_bound[aloc\_i])break;

if (all > cumsum[aloc\_i]&&useful\_bound[aloc\_i]) {

drop = true;

break;

//count++;

}

}\*/

//if (count == week - i-1||count == 0)drop = false;

//else drop = true;

//检测有没有库存超限，超限情况下应当另外讨论，这里为了简化模型选择直接放弃此次分配

int minist = 6000;

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

minist = min(block[aloc\_i][no][3], minist);

if (block[aloc\_i][no][3] <=0){//< a[i][pro]) {

drop = true;

break;

}

}

if (drop) break;

minist = min(minist, (int)((capacity\_need - weekly\_actual\_capacity[week])\*actual\_consume[kind-1]));

minist = max(minist, 0);

//必要的时候分割转运操作避免爆仓

if (minist > 0 && minist < a[i][pro]) {

//cout << a[i][pro]<<" ";

a[i][pro] -= minist;

int tmp = minist;

minist = a[i][pro];

a[i][pro] = tmp;

//cout << a[i][pro] << endl;

}

else minist = 0;

//cout << a[i][pro] << endl;

//通过检测则放入

weekly\_actual\_capacity[week] += (1. - 0.01\*b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1])+0.5;//(1-损耗率）\*总量/单位消耗

ave += (1. - 0.01 \* b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1]);

//cout << weekly\_actual\_capacity[week] << endl;

//在[i,week]期间，将货物存放到no号仓库内，更新时空图和仓储分区与实际产能,校验累积和边界

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

cumsum[aloc\_i] -= a[i][pro] / (actual\_consume[kind - 1]);

block[aloc\_i][no][kind-1] += a[i][pro];

block[aloc\_i][no][3] -= a[i][pro];

allocate\_status\_map[aloc\_i][no][kind - 1].push(make\_pair(make\_pair(i, pro),a[i][pro]));

//cout << i << " " << pro << " " << a\_[i][pro] << endl;

}

allocat\_ans[i][pro] += a[i][pro];

//清空供货商相关存量

a[i][pro] = minist;

}

}

}

}//with\_bound

for (int i = 0; i < useful\_bound.size(); i++) {

if (weekly\_actual\_capacity[i] >= capacity\_need - 1) {

useful\_bound[i] = true;

}

else {

useful\_bound[i] = false;

//cout << i << endl;

}

}

capacity\_need \*= 2;//获取超出的材料

steps = optical\_trans\_order(weeks, 8);

//sort(steps.begin(), steps.end(), [](auto& x, auto& y) {return (x.first < y.first); });

for (auto& step : steps) {

int week = step.first;//获取周数

int no = step.second.first;//获取仓库号

for (int i = week; i >= 0; i--) {

for (int kind = 1; kind < 4; kind++) {//倒序依次装配A B C货物

for (int pro = 0; pro < a[week].size(); pro++) {//其实可以优化一下，减少外部碎片，这里暂时先不管了

if (kind\_dict[pro] != kind || a[i][pro] == 0)continue;

if (useful\_bound[week])continue;

bool drop = false;

//检测是否有必要扩

//检测有没有库存超限，超限情况下应当另外讨论，这里为了简化模型选择直接放弃此次分配

int minist = 6000;

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

minist = min(block[aloc\_i][no][3], minist);

if (block[aloc\_i][no][3] <= 0) {//< a[i][pro]) {

drop = true;

break;

}

}

//cout << minist << endl;

if (drop) break;

minist = min(minist, (int)((capacity\_need - weekly\_actual\_capacity[week]) \* actual\_consume[kind - 1]));

minist = max(minist, 0);

//必要的时候分割转运操作避免爆仓

if (minist > 0 && minist < a[i][pro]) {

//cout << a[i][pro]<<" ";

a[i][pro] -= minist;

int tmp = minist;

minist = a[i][pro];

a[i][pro] = tmp;

//cout << a[i][pro] << endl;

}

else minist = 0;

//cout << a[i][pro] << endl;

//通过检测则放入

weekly\_actual\_capacity[week] += (1. - 0.01 \* b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1])+0.5;//(1-损耗率）\*总量/单位消耗

ave += (1. - 0.01 \* b[week][no]) \* a[i][pro] / (actual\_consume[kind - 1]);

if (weekly\_actual\_capacity[week] >= capacity\_need / 2 - 1)useful\_bound[week] = true;

//在[i,week]期间，将货物存放到no号仓库内，更新时空图和仓储分区与实际产能,校验累积和边界

for (int aloc\_i = i; aloc\_i <= week; aloc\_i++) {

cumsum[aloc\_i] -= a[i][pro] / (actual\_consume[kind - 1]);

block[aloc\_i][no][kind - 1] += a[i][pro];

block[aloc\_i][no][3] -= a[i][pro];

allocate\_status\_map[aloc\_i][no][kind - 1].push(make\_pair(make\_pair(i, pro), a[i][pro]));

//cout << i << " " << pro << " " << a\_[i][pro] << endl;

}

allocat\_ans[i][pro] += a[i][pro];

//清空供货商相关存量

a[i][pro] = minist;

}

}

}

}//without bound

cout << "周产能:";

int sum = 0;

for (auto& e : weekly\_actual\_capacity) {

cout << e << " ";

sum += e;

}

cout << endl;

cout << ave/24 << endl;

cout << "转运决策如下：";

sort(steps.begin(), steps.end(), [](auto& x, auto& y) {return x.first < y.first; });

//cout << "week1:" << endl;

int nums = 8;

for (int i = 0; i < steps.size(); i++) {

if (i % nums == 0)cout <<"第"<<steps[i].first<<"周"<< endl;

if (block[i/nums][steps[i].second.first][3] == 6000)continue;

cout <<"仓库" << steps[i].second.first <<" 损耗率"<< steps[i].second.second<<" 运送"<< endl;

cout << "仓储分区情况：" << endl;

cout << "A:" <<block[i/nums][steps[i].second.first][0]<< " B:"<< block[i / nums][steps[i].second.first][1] << " C:" << block[i / nums][steps[i].second.first][2] << " empty:" << block[i / nums][steps[i].second.first][3] << endl;

cout << "具体策略如下：" << endl;

int e = steps[i].second.first;

for (int j = 0; j < 3; j++) {

cout << "材料种类" << j << endl;

while (!allocate\_status\_map[i / nums][e][j].empty()) {

auto& tmp = allocate\_status\_map[i / nums][e][j].top();

allocate\_status\_map[i / nums][e][j].pop();

cout << "存入周:" << tmp.first.first << "\t商家:" << tmp.first.second << "\t 数量" << tmp.second <<'\t'<< endl;

//allocat\_ans[tmp.first.first][tmp.first.second] += tmp.second;

}

}

}

cout << "订货情况：（每行 为一个商家的数据）"<<endl;

for (int i = 0; i < a[0].size(); i++) {

for (int j = 0; j < a.size(); j++) {

cout << allocat\_ans[j][i]<< '\t';

}

cout << endl;

}

cout << "材料空闲情况" << endl;

for (int i = 0; i < a[0].size(); i++) {

for (int j = 0; j < a.size(); j++) {

cout << (-allocat\_ans[j][i] + a\_[j][i])<< '\t';

}

cout << endl;

}

return true;

}

};

int main(void) {

Solution s = Solution();

s.execute();

char ch;

cin >> ch;

return 0;

}