1. Исходные данные:

Рассматривается цех по производству макаронных изделий. В цехе две одинаковые производственные линии. Номенклатура выпускаемых изделий составляет 10 наименований.

Таблица 1. Номенклатура продукции

Наименование товара	**Производительность (кг/час)**	**План на месяц (кг)**	**Остаток на складе на начало месяца (кг)**
Рожки	1100	256995	25670
Перья	1160	290350	33830
Рожки вит.	1100	247850	16400
Маргаритка	1000	9980	3550
Пуговка	1050	6678	3600
Вермишель	1000	180660	24550
Серпантин	1150	92200	7800
Сапожок	1150	231950	22780
Ракушка	1160	22200	3600
Колечко	1150	34800	5260

Во второй графе таблицы 1 приведена производительность каждой из линий по каждому виду продукции (кг/час), в третьей графе план производства на месяц (кг) в соответствии со сформированными заявками покупателей, в четвертой графе – остатки готовой продукции на складе на начало месяца.

Стоимость 1 кг любой макаронной продукции равна 20 руб. Издержки хранения 1 тонны макаронной продукции в течение суток равны 15 рублям.

При переходе работы линии с одного вида продукции на другой происходит смена матрицы. Время на замену матрицы равно одному часу, кроме того. При смене матрицы происходит потеря 100 кг продукции (технологический брак). С учетом этих потерь и затрат на переналадку, стоимость одной переналадки равна 3000 рублей.

В таблице 2 представлен план поставки продукции потребителям, сформированный в отделе сбыта. Пi –номер потребителя (всего 11 потребителей). В скобках указана дата месяца, на которую запланирована отгрузка продукции. Например, потребителю №3 запланирована поставка на 12 и 28 число месяца. Если даты поставки не указаны (как, например у

потребителя № 2), то предполагается, что поставка может осуществляться в любой день месяца. За несвоевременную поставку продукции предусмотрены штрафные санкции в размере 5 процентов общей стоимости поставки за каждый день просрочки.

Таблица 2. План поставки продукции потребителям

	П1(03)	**П1(13)**	**П1(23)**	**П2**	**П3(12)**	**П3(28)**	**П4(10)**	**П4(20)**	**П4(30)**	**П5**
Товар										
Рожки	22500	22500	22500	23650		12100				
Перья					25100	25100	14500	14500	14500	16250
Рожки вит.	16500	16500	16500	14300	13150	13150	12200	12200	12200	15500
Маргаритка				9980						
Пуговка	2226	2226	2226							
Вермишель				12200			12100	12100	12100	
Серпантин	4600	4600	4600		6500	6500				
Сапожок	12450	12450	12450	14400			12200	12200	12200	12600
Ракушка				9900						
Колечко				9800						

Таблица 2. План поставки продукции потребителям (продолжение таблицы 2)

	П6(5)	**П6(10)**	**П6(15)**	**П6(20)**	**П6(25)**	**П6(30)**	**П7**
Товар							
Рожки	8200	8200	8200	8200	8200	8200	22500
Перья	8200	8200	8200	8200	8200	8200	
Рожки вит.	10100	10100	10100	10100	10100	10100	12150
Маргаритка							
Пуговка							
Вермишель	8200	8200	8200	8200	8200	8200	12200
Серпантин	8200	8200	8200	8200	8200	8200	
Сапожок	8200	8200	8200	8200	8200	8200	15500
Ракушка							
Колечко	3200	3200	3200	3200	3200	3200	

Таблица 2. План поставки продукции потребителям (продолжение таблицы 2)

	П8(10)	П8(20)	П8(30)	П9(6)	П9(25)	П10	П11(4)	П11(15)	П11(26)
Товар									
Рожки				12200	12200	15600	14015	14015	14015
Перья	11065	11065	16320	14200	14200	15600	16250	16250	16250
Рожки вит.				8600	8600	15700			
Маргаритка									
Пуговка									
Вермишель	12200	12200	12200	12200	12200	9760			
Серпантин							5400	5400	5400
Сапожок	14100	14100	14100			12600	3800	3800	3800
Ракушка	4100	4100	4100						
Колечко						5800			

Требуется разработать почасовой график работы линий в течение месяца (работа в три смены, 24 часа в сутки) и соответствующие построенному графику суммарные месячные затраты (на хранение остатков продукции, переналадку и штрафы от несвоевременной поставки).

2. Выполнение работы:

3.1 Построение математической модели и алгоритмов

Для начала нам нужно определить оптимальный размер партии исходя из того, что нужно минимизировать издержки на хранение и переналадку линий.

C1 - Затраты на наладку

C2 - Издержки на хранение единицы продукции в месяц

Q - План на месяц (потребность в месяц)

X - Оптимальная партия

$$\frac{Q}{X} * C1 + \frac{X}{2} * C2 \rightarrow min$$

Возьмем производную от этой функции:

$$-\frac{Q * C1}{X^2} + \frac{C2}{2} = 0$$

По итогу мы получаем формулу Вильсона по которой мы определяем оптимальную партию в кг.:

$$X = \sqrt{\frac{2 * C1 * Q}{C2}}$$

При этом C1 и C2 будут равны для всех товаров:

C1 = 3000,

C2 = 0,225 ((15руб * 30 дней) / (1000 кг * 2))

Чтобы найти за какое время изготовляется оптимальная партия, мы Полученную оптимальную партию в кг. должны разделить на производительность производственной линии:

Хопт = X / P, где P - производительность одной линии.

3.2 Разработка ПО

Вначале нужно разместить исходные данные в корректном порядке поставки поставщикам. Сверху таблицы указаны числа, эти числа и есть числа отправки.

			п1	п11	п6	п9		п2	п5	п4+п6 + п8		п3	п1		п6+п11
Дата/часы	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Рожки			22500	14015	8200	12200				8200			22500		22215
Перья			0	16250	8200	14200				33765		25100			24450
Рожки вит.			16500		10100	8600				22300		13150	16500		10100
Маргаритка			0							0					0
Пуговка			2226							0			2226		0
Вермишель			0		8200	12200				32500					8200
Серпантин			4600	5400	8200					8200		6500	4600		13600
Сапожок			12450	3800	8200					34500			12450		12000
Ракушка			0							4100					0
Колечко			0		3200					3200					3200

Рисунок 1.1 – Данные в верном порядке

	п7	п10		п4+п6+п8			п1		п6+п9	п11		п3		п4+п6+п8
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
				8200			22500		20400	14015		12100	61750	8200
				33765					22400	16250		25100	31850	39020
				22300			16500		18700			13150	57650	22300
				0					0				9980	0
				0			2226		0				0	0
				32500					20400				34160	32500
				8200			4600		8200	5400		6500	0	8200
				34500			12450		8200	3800			55100	34500
				4100					0				9900	4100
				3200					3200				15600	3200

Рисунок 1.2 – Данные в верном порядке. Продолжение.

mine.py

```
import math
import numpy as np
```

Те данные, которые мы упорядочили в excel, вводим в программу через массив.

```
a = np.array([[…]])
arrayNeeds = a.transpose()
x, power, newPertia = 0, 0, 0

listX = []  # оптимальная партия
C1 = 3000  # стоимость наладки
C2 = 0.450  # затраты на хранения за тонну в мес
listQ = [256995, 290350, 247850, 9980, 6678, 180660, 92200, 231950,
22200, 34800]  # потребность в месяц
listOst = [25670, 33830, 16400, 3550, 3600, 24550, 7800, 22780, 3600,
5260]  # остаток на начало месяца
listOst1 = listOst.copy()
listOst2 = listOst.copy()
listPower = [1100, 1160, 1100, 1000, 1050, 1000, 1150, 1150, 1160,
1150]  # производительность
listOptTime = []
listOptPartia = []
```

Расчет оптимальной производимой партии (формула Вильсона).

```
for i in range(len(listQ)):
    x = ((2 * listQ[i] * C1) / C2) ** (1 / 2)
    x = round(x)
    listX.append(x)
```

Расчет затраченного времени на производство оптимальной партии.

```
for i in range(len(listX)):
    power = listX[i] / listPower[i]
    listOptTime.append(power)
```

Оптимальная партия производится не целое количество часов, поэтому мы округляем часы производства в большую сторону и партия немного увеличивается.

```
for i in range(len(listX)):
    newPartia = math.ceil(listOptTime[i]) * listPower[i]
    listOptPartia.append(newPartia)
    listOptTime[i] = math.ceil(listOptTime[i])
```

Для выстраивания очереди на производственных линиях смотрим какой продукт потребуется для отгрузки раньше с учетом первоначальных остатков. Потребность определяется датой отгрузки партии поставщикам.

```
ostatok = 0
listOchered = []
newList = []

for i in range(30):  # столбец
    for j in range(0, 10):  # строка
        ostatok = listOst[j] - a[j][i]
```

```
        listOst[j] = ostatok
        if ostatok < 0:
            newList.append(j)
```

Выстраиваем общую очередь с учетом потребности в продукте.

```
for i in newList:  # выстраиваем очередь
    if i not in listOchered:
        listOchered.append(i)

hour1 = 1
hour2 = 1
line1 = []
line2 = []
listProverit1 = []
listProverit2 = []


print("Лист очереди ----", listOchered)
```

Распределяем общую очередь изготовления макаронных изделий на две производственные линии в зависимости от потребностей покупателей, времени изготовления продукта и загруженности линии. При переходе на изготовление другого вида макарон мы добавляем один час на переналадку линии.

```
for i in listOchered:
    if listOst1[i] < listQ[i]:

        if (len(listOchered) - 10) == 0:
            listOst1[i] += listOptPartia[i]
            hour1 += listOptTime[i]
            hour1 += 1  # переналадка
            line1.append(hour1)
            listProverit1.append(i)
            listOchered.append(i)

        else:
            if (hour1 - hour2) >= 0:
                listOst1[i] += listOptPartia[i]
                hour2 += listOptTime[i]
                hour2 += 1  # переналадка
                line2.append(hour2)
                listProverit2.append(i)
                listOchered.append(i)
            else:
                listOst1[i] += listOptPartia[i]
                hour1 += listOptTime[i]
                hour1 += 1  # переналадка
                line1.append(hour1)
                listProverit1.append(i)
                listOchered.append(i)

    else:
        continue
```

Производим товар по часам работы компании в течение месяца на двух линиях.

```
listOst3 = []
zatrat = 0
count = 0
count2 = 0
sum1 = 0
for hour in range(1, 720): # в месяце 720 часов
   if count < len(line1) and count2 < len(line2):
       listOst2[listProverit1[count]] +=
listPower[listProverit1[count]]
       listOst2[listProverit2[count2]] +=
listPower[listProverit2[count2]]
       print(hour, 'час ----', listOst2[listProverit1[count]])
       print(listOst2)

       if hour == line1[count] and hour !=  line1[len(line1) - 1]:
           count += 1
           print('Индекс товара 1 ==== ', count)

       if hour == line2[count2] - 1:
           count2 += 1
           print('Индекс товара 2 ==== ', count2)
```

Отгружаем товары по дням в соответствии с графиком отгрузки. Рассчитываем затраты на хранение продукции на складе и начисляем штраф за несвоевременную отгрузку товара покупателям (если такое происходит).

```
       if hour != 0 and hour % 24 == 0:
           day = hour//24
           print("день-----> ",day)
           for i in range(day, day + 1):
               for j in range(0, 10):
                   if listOst2[j] >= a[j][i]:
                       listOst3.append(listOst2[j] - a[j][i])
                   else:
                       for x in range(0, 10):
                           sum1 += a[x][i]
                       listOst3.clear()
                       break
               if len(listOst3) != 0:
                   listOst2 = listOst3.copy()
                   listOst3.clear()
               zatrat = zatrat + sum1 * 20 * 0.05
               zatrat = zatrat + (sum(listOst2) / 1000) * 15
       print(zatrat)
   else:
       break
```

Рассчитываем затраты на переналадку каждой линии.

```
zatrat = zatrat + (len(line1) - 1) * 3000 + len(line2) * 3000
```

Выводим рассчитанные значения.

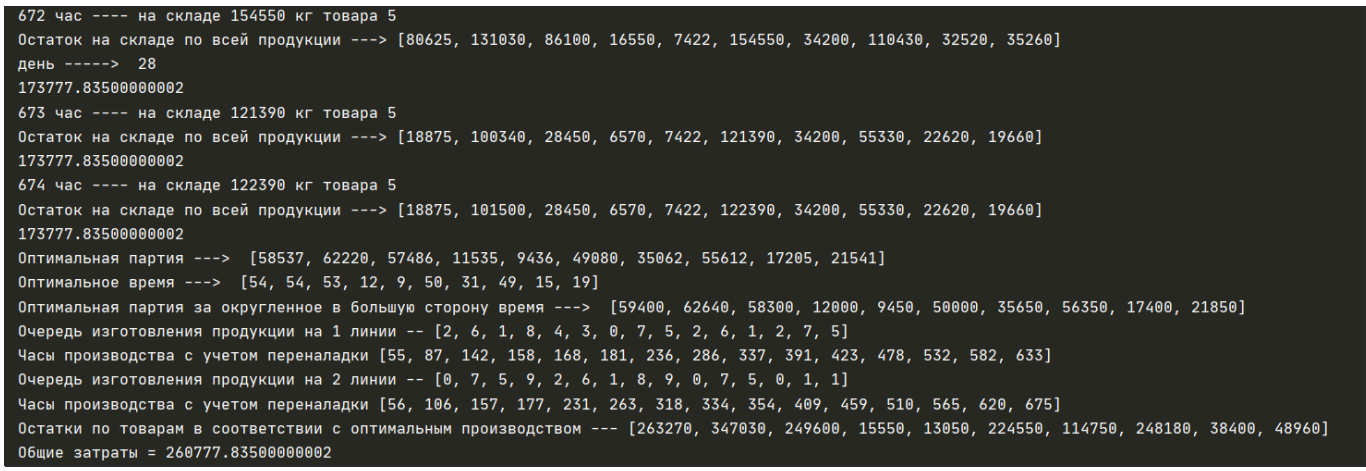

Рисунок 2 - Вывод полученных значений

grafic.py

Для построения графика импортируем необходимые библиотеки для построения и данные о длительности производственных линий. После этого присвоим каждому виду изготавливаемой продукции цвет, чтобы они отличались на графике. Далее строим прямые в соответствии с полученными данными, отмечаем начало работы после каждой смены матрицы вертикальными пунктирными линиями и выводим на экран. При желании библиотеки позволяют изменять масштаб графика для детального рассмотрения информации.

Для обозначения продукции были использованы следующие цветовые значения:

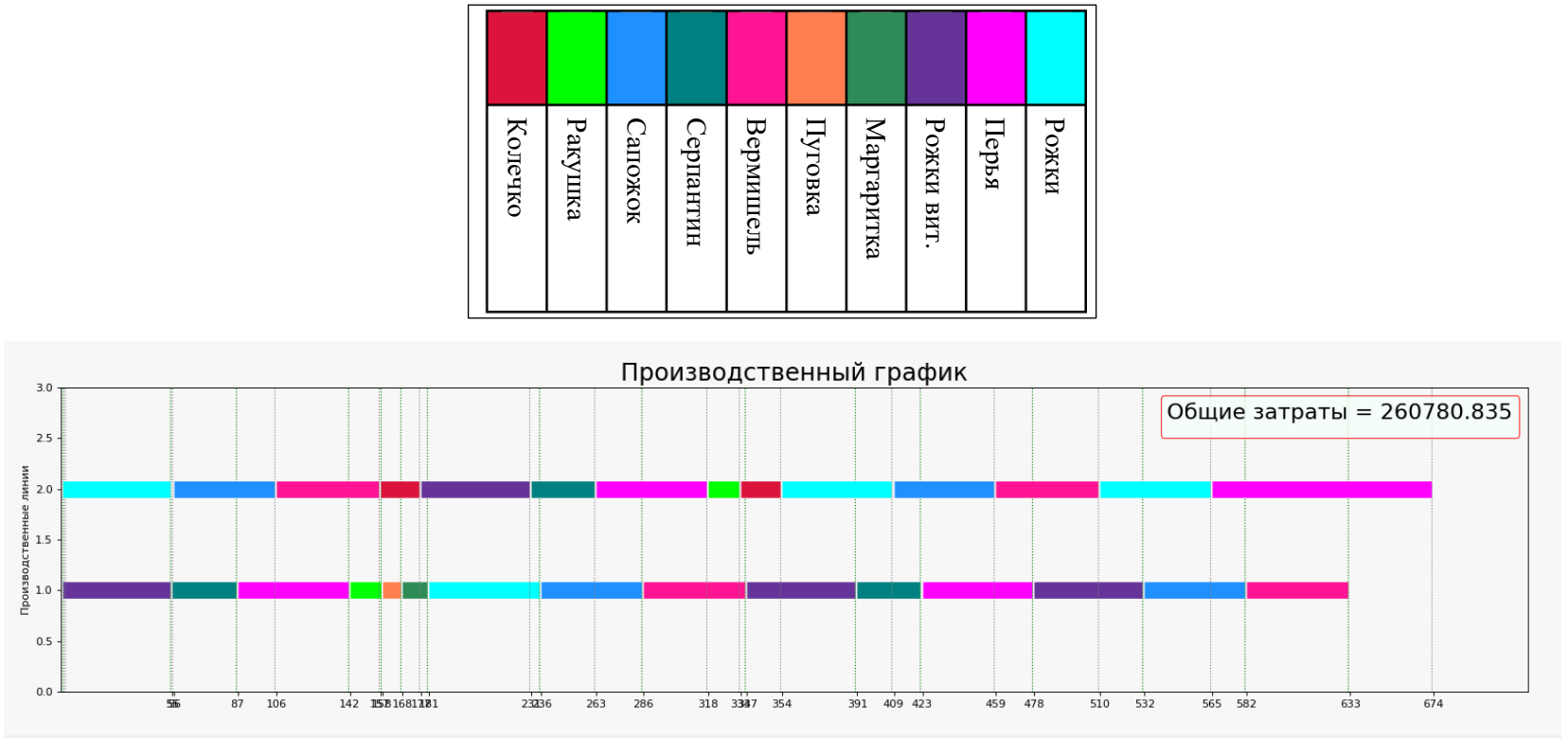

Рисунок 3.1 - Почасовой график работы линий

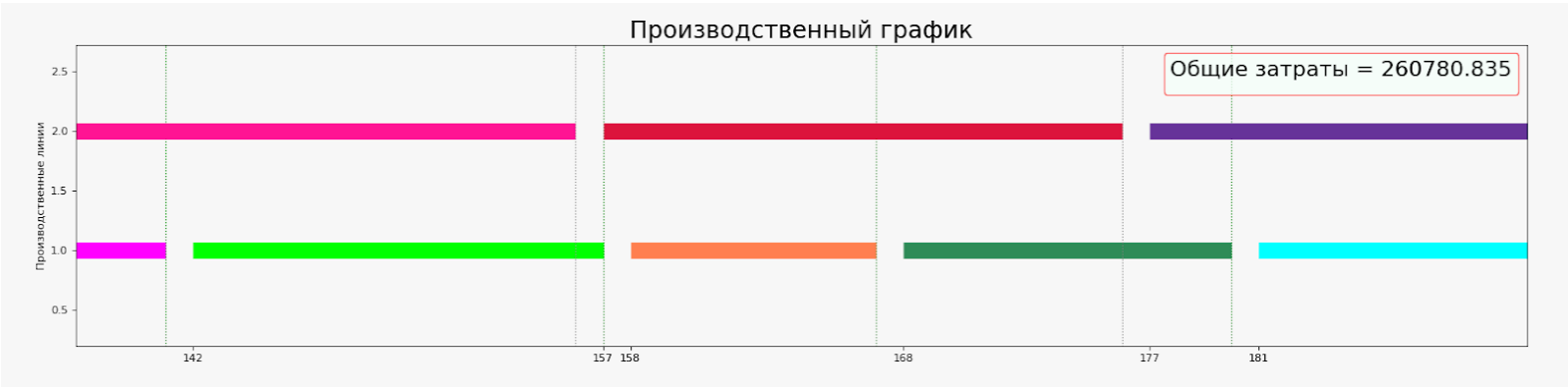

Рисунок 3.2 - Сегмент почасового графика работы линий

Заключение

В ходе расчетного задания разработан почасовой график работы линий в течение месяца (работа в три смены, 24 часа в сутки) и рассчитаны соответствующие построенному графику суммарные месячные затраты (на хранение остатков продукции, переналадку и штрафы от несвоевременной поставки). Благодаря рассчитанным оптимальным партиям производство продукции происходит 28 дней вместо 30 из-за чего общие затраты составили 260780,84 рубля.