Министерство науки и высшего образования Российской Федерации

Государственное образовательное учреждение высшего профессионального образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра Прикладная математика

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель      А. В. Сорокин

(подпись) (и.о.фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

дата

Отчет

по дисциплине

ОСНОВЫ МОДЕЛИРОВАНИЯ

лабораторная работа №1

Модели Джонсона задач упорядочения nx2 и nx3

название работы

ЛР 09.03.04.05.001 О

обозначение документа

Студент группы гр. ПИ-92                                              Д. А. Савиков

*(подпись*)

Преподаватель доцент, к.т.н.         А. В. Сорокин

должность, ученое звание и.о., фамилия

БАРНАУЛ 2022

# **Аннотация**

В данной лабораторной работе были рассмотрены задачи упорядочения nx2 и nx3 из теории расписаний. Были изучены различные модели и алгоритмы нахождения оптимального решения. Также был изучен вопрос о полном переборе вариантов решения задач упорядочения для поиска оптимального решения, в случае не выполнения необходимого условия сведения задачи nx2 к nx3. Кроме того, был построен график Ганта.

# **Оглавление**

[**Аннотация**](#_fy8tgb9wrb4h)2

[**Оглавление**](#_bnuld8wplxsh)3

[**Задание 1. Задача упорядочивания nx2**](#_mmjjbivral23)4

[Решение задачи задания 1](#_s1f9aspa2odj) 5

[Алгоритм решения:](#_pbqn30dd12xe) 5

[Код программы на языке С#](#_ju4cfcd8b951) 5

[Подключенные библиотеки:](#_io0be01heg68) 6

[Код Main:](#_u36aa3anqv40) 6

[Код функции вывода таблицы nx2:](#_9y4jvn74z6xv) 8

[Код вывода диаграммы Ганта для nx2:](#_hn253gg3nopv) 9

[Код оптимизации:](#_ygsy0ahvhu50) 9

[Результаты выполнения задания №1](#_oguxdwehax74) 11

[**Задание 2. Задача упорядочивания nx3**](#_5ejle55b4u7h)12

[Решение задачи задания 2](#_ymxnz44ib12) 13

[Алгоритм решения:](#_wbglp2lslw4u) 13

[Код программы на языке С#](#_no4p4m7ukw7a) 13

[Код Main:](#_14oqwcnnewwt) 13

[Код функции вывода таблицы nx3:](#_8w81bn6z8mom) **Ошибка! Закладка не определена.**

[Код вывода диаграммы Ганта для nx3:](#_rf0q5ub1b4jt) **Ошибка! Закладка не определена.**

[Код нахождения Т для текущих значений:](#_n3b9ta94ed71) **Ошибка! Закладка не определена.**

[Код оптимизации:](#_87ngn3l3bthi) 19

[Результаты выполнения задания №2](#_o1me0xeqb2tp) 20

[**Задание 3. Задача упорядочивания nx3 решаемая методом перебора**](#_7ygtjltn6lld)21

[Решение задачи задания 3](#_apqz3f9mqwba) 21

[Алгоритм решения:](#_an4kbjatzy0r) 21

[Код программы на языке С#](#_x7ucxv4ltse1) 22

[Код Main:](#_j6cojmqjanzv) 28

[Код функции вывода таблицы nx3:](#_oiv5jrqrr6xi) 28

[Код вывода диаграммы Ганта для nx3:](#_4osl8hab9q6m) **Ошибка! Закладка не определена.**

[Код нахождения Т для текущих значений:](#_juvk7taetdv1) **Ошибка! Закладка не определена.**

[Код нахождения всех перестановок:](#_shlhjhi0pzgf) **Ошибка! Закладка не определена.**

[Результаты выполнения задания №3](#_lyzmphgd17sa) 28

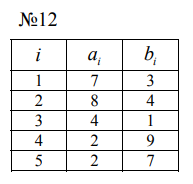
[**Заключение**](#_xf1bjlhgvqvp)29

[**Список используемых источников**](#_2bn6wsx)31

# **Задание 1. Задача упорядочивания nx2**

1. Разобраться с построением модели задачи Джонсона‐Белмана nx2 из теории расписаний.
2. Ознакомиться с формулировкой задачи Джонсона‐Белмана nx2.
3. Научиться строить график Ганта для задачи Джонсона‐Белмана nx2 (с n деталями и двумя станками).
4. Изучить процесс построения модели задачи Джонсона (nx2) на основе графика Ганта.
5. Изучить возможности поиска оптимального решения задачи Джонсона для двух станков. Изучить подход вывода алгоритма Джонсона и сам алгоритм. Решить задачу упорядочения согласно варианту с использованием алгоритма Джонсона, найдя оптимальный порядок запуска последовательности *n* деталей в обработку на двух станках.
6. Построить график Ганта для оптимальной последовательности запуска деталей своей задачи согласно варианту.
7. Рассчитать оптимальное время окончания обработки всех деталей на двух станках. Это удобно сделать, перенумеровывая оптимальный порядок в порядке возрастания индексов и затем использовать формулы из пункта 4. График Ганта тоже можно приводить с новой нумерацией и новым определением цвета.

**Вариант 12:**



## Решение задачи задания 1

### **Алгоритм решения:**

Был создан класс Tabel, объект которого хранит номер одной детали, время a, b, и c (используется в 2 и 3 задании). Таблица предствлена объектом класса List<Tabel>.

Оптимизация происходит по алгоритму: цикл проходит по всем строкам таблицы (кроме последней) и ищет минимальное число среди , , и . Если минимальным оказывается или , то ничего не происходит. Иначе строки i и i+1 меняются местами. После этого на последнем месте стоит нужная строка и необходимо повторять этот алгоритм пока не будет оптимизирована таблица, но в каждом последующем цикле необходимо проверять на одну меньше строк чем в предыдущем цикле.

После оптимизации на экран выводятся: оптимальная таблица, последовательность запуска, диаграмма Ганта и время обработки деталей. Время высчитывается параллельно с построением диаграммы Ганта. Каждый простой на станке B высчитывается по формуле:

## Код программы на языке С#

Фрагмент, относящийся к Заданию №1.

Все три задачи были реализованы в одном проекте. Если в представленном фрагменте есть код, относящийся к другой задаче, он будет выделен зеленым фоном.

### **Подключенные библиотеки:**

Данный фрагмент относиться ко всем задачам и будет приведён только в задаче №1

using System;

using System.Collections.Generic;

using System.Linq;

using System.IO;

### **Код описания класса Table**

### public class Tabel

### {

### public int num;

### public int a;

### public int b;

### public int c;

### public Tabel()

### {

### }

### public Tabel(int num, int a, int b)

### {

### this.num = num;

### this.a = a;

### this.b = b;

### }

### public Tabel(int num, int a, int b, int c)

### {

### this.num = num;

### this.a = a;

### this.b = b;

### this.c = c;

### }

### }

### **Код сортировки массива:**

### public static void Swap(List<Tabel> tab, int i, int j)

### {

### Tabel mid;

### mid = tab[j];

### tab[j] = tab[i];

### tab[i] = mid;

### }

### public static void NX2(List<Tabel> tab)

### {

### int[] help = new int[4];

### int min;

### for(int i = 0; i < tab.Count - 1; i++)

### {

### for(int j = 0; j < tab.Count - i - 1; j++)

### {

### help[0] = tab[j].a;

### help[1] = tab[j].b;

### help[2] = tab[j + 1].a;

### help[3] = tab[j + 1].b;

### min = help.Min();

### if(min == help[1] || min == help[2])

### {

### Swap(tab, j, j + 1);

### }

### }

### }

### }

### **Код вывода таблицы:**

### public static void ShowTabel(List<Tabel> tab, int type)

### {

### if(type == 1)

### {

### Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

### Console.WriteLine(" № \* a \* b ");

### Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

### }

### else

### {

### Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

### Console.WriteLine(" № \* a \* b \* c ");

### Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

### }

### for(int i = 0; i < tab.Count; i++)

### {

### if(type == 1)

### {

### Console.WriteLine(tab[i].num.ToString().PadLeft(5) + "\*" + tab[i].a.ToString().PadLeft(5) + "\*" +

### tab[i].b.ToString().PadLeft(5));

### Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

### }

### else

### {

### Console.WriteLine(tab[i].num.ToString().PadLeft(5) + "\*" + tab[i].a.ToString().PadLeft(5) + "\*" +

### tab[i].b.ToString().PadLeft(5) + "\*" + tab[i].c.ToString().PadLeft(5));

### Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

### }

### }

### }

### **Код вывода Диаграммы Ганта:**

### public static int Gant(List<Tabel> tab, int type)

### {

### Console.Write("A: ");

### for(int i = 0; i < tab.Count; i++)

### {

### Console.Write(string.Concat(Enumerable.Repeat(tab[i].num, tab[i].a)));

### }

### Console.WriteLine();

### Console.Write("B: ");

### int Ta, Tb;

### Ta = Tb = 0;

### for(int i = 0; i < tab.Count; i++)

### {

### Ta += tab[i].a;

### if(Ta - Tb > 0)

### {

### Console.Write(string.Concat(Enumerable.Repeat("x", Ta - Tb)));

### Tb += Ta - Tb;

### }

### Console.Write(string.Concat(Enumerable.Repeat(tab[i].num, tab[i].b)));

### Tb += tab[i].b;

### }

### if(type == 2)

### {

### Console.WriteLine();

### Console.Write("C: ");

### int Tc;

### Ta = Tb = Tc = 0;

### for (int i = 0; i < tab.Count; i++)

### {

### Ta += tab[i].a;

### if (Ta - Tb > 0)

### {

### Tb += Ta - Tb;

### }

### Tb += tab[i].b;

### if (Tb - Tc > 0)

### {

### Console.Write(string.Concat(Enumerable.Repeat("y", Tb - Tc)));

### Tc += Tb - Tc;

### }

### Console.Write(string.Concat(Enumerable.Repeat(tab[i].num, tab[i].c)));

### Tc += tab[i].c;

### }

### Console.WriteLine();

### return Tc;

### }

### Console.WriteLine();

### return Tb;

### }

### **Код Main:**

## static void Main(string[] args)

## {

## StreamReader stream = new StreamReader(Console.ReadLine());

## string line;

## string[] data = new string[5];

## int help;

## List<Tabel> NX2Tabel = new List<Tabel>();

## List<Tabel> NX3Tabel = new List<Tabel>();

## List<Tabel> PereborTabel;

## line = stream.ReadLine();

## while(line != "")

## {

## data = line.Split(' ');

## NX2Tabel.Add(new Tabel(Convert.ToInt32(data[0]), Convert.ToInt32(data[1]), Convert.ToInt32(data[2])));

## line = stream.ReadLine();

## }

## while((line = stream.ReadLine()) != null)

## {

## data = line.Split(' ');

## NX3Tabel.Add(new Tabel(Convert.ToInt32(data[0]), Convert.ToInt32(data[1]), Convert.ToInt32(data[2]), Convert.ToInt32(data[3])));

## }

## stream.Close();

## PereborTabel = new List<Tabel>(NX3Tabel);

## Console.WriteLine("1 задание");

## Console.WriteLine("Исходная таблица");

## ShowTabel(NX2Tabel, 1);

## Console.WriteLine("Диаграмма Ганта");

## help = Gant(NX2Tabel, 1);

## Console.WriteLine("Время = " + help);

## NX2(NX2Tabel);

## Console.WriteLine("Оптимальная последовательность");

## ShowWay(NX2Tabel);

## Console.WriteLine("Таблица");

## ShowTabel(NX2Tabel, 1);

## Console.WriteLine("Диаграмма Ганта");

## help = Gant(NX2Tabel, 1);

## Console.WriteLine("Время = " + help);

………

}

## Результаты выполнения задания №1

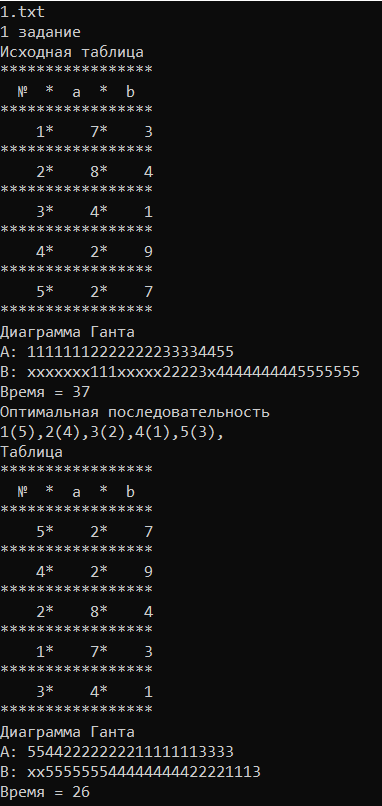
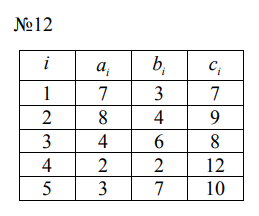


Рис 1. Исходная таблица значений; исходный технологический маршрут, график Ганта, построенный на исходных данных, исходное полное время работы; оптимальная таблица; оптимальный технологический маршрут, график Ганта, оптимальное время обработки деталей.

# **Задание 2. Задача упорядочивания nx3**

1. Разобраться с построением модели задачи Джонсона‐ Белмана nx3 из теории расписаний.
2. Ознакомиться с формулировкой задачи Джонсона‐ Белмана nx3.
3. Научиться строить график Ганта для задачи Джонсона‐Белмана с n деталями и тремя станками.
4. Изучить процесс построения модели задачи Джонсона с (nx3) на основе графика Ганта.
5. Изучить возможности поиска оптимального решения задачи Джонсона для трех станков.

**Вариант 12:**



## Решение задачи задания 2

### **Алгоритм решения:**

Сначала происходит проверка условия:

Если истина, то задача сводится к решению алгоритмом Джонсона. Происходит формирование новой таблицы nx2, по правилу:



Далее к сформированной таблице nx2 применяется метод Джонсона, описанный в реализации 1-го задания. После оптимизации таблицы nx2 происходят необходимые перестановки в исходной таблице nx3, чтобы номера строк таблиц nx2 и nx3 совпадали.

График Ганта в данном случае для первых двух станков выводится аналогично первому заданию, в случае с третьим станком, простои вычисляются по формуле (код в 1 задании):

## Код программы на языке С#

Фрагмент, относящийся к Заданию №2.

### **Код сортировки массива:**

### public static void NX3(List<Tabel> tab)

### {

### int minA = tab.Min(tabel => tabel.a), maxB = tab.Max(tabel => tabel.b), minC = tab.Min(tabel => tabel.c);

### Console.WriteLine("min a = " + minA + " max b = " + maxB + " min c = " + minC);

### if(minA >= maxB || minC >= maxB)

### {

### Console.WriteLine("Условие (min a >= max b) || (min c >= max b) выполнено");

### List<Tabel> twoTab = new List<Tabel>();

### for(int i = 0; i < tab.Count; i++)

### {

### twoTab.Add(new Tabel(tab[i].num, tab[i].a + tab[i].b, tab[i].b + tab[i].c));

### }

### NX2(twoTab);

### int index;

### for(int i = 0; i < tab.Count; i++)

### {

### index = twoTab.FindIndex(x => x.num == tab[i].num);

### Swap(tab, i, index);

### if (i != index) i--;

### }

### }

### else

### {

### Console.WriteLine("Условие (min a >= max b) || (min c >= max b) невыполнено");

### }

### }

### **Код вывода таблицы:**

Описан в 1 задании

### **Код вывода Диаграммы Ганта:**

Описан в 1 задании

### **Код Main:**

static void Main(string[] args)

{

………

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//2 задание

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Console.WriteLine("2 задание");

Console.WriteLine("Исходная таблица");

ShowTabel(NX3Tabel, 2);

Console.WriteLine("Диаграмма Ганта");

help = Gant(NX3Tabel, 2);

Console.WriteLine("Время = " + help);

NX3(NX3Tabel);

Console.WriteLine("Оптимальная последовательность");

ShowWay(NX3Tabel);

Console.WriteLine("Таблица");

ShowTabel(NX3Tabel, 2);

Console.WriteLine("Диаграмма Ганта");

help = Gant(NX3Tabel, 2);

Console.WriteLine("Время = " + help);

………

}

## Результаты выполнения задания №2

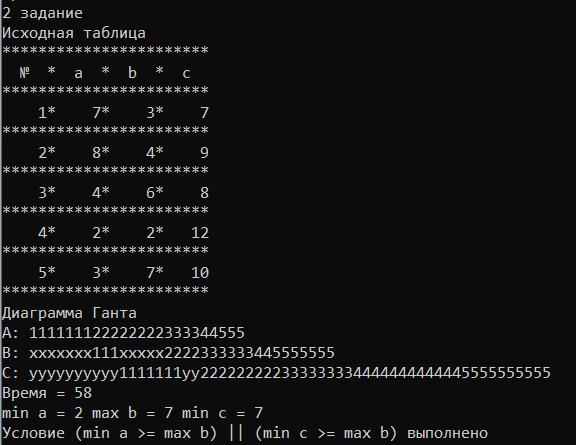


Рис 1. Исходная таблица значений, исходный маршрут, диаграмма Ганта, текущее время работы и проверка условия сводимости

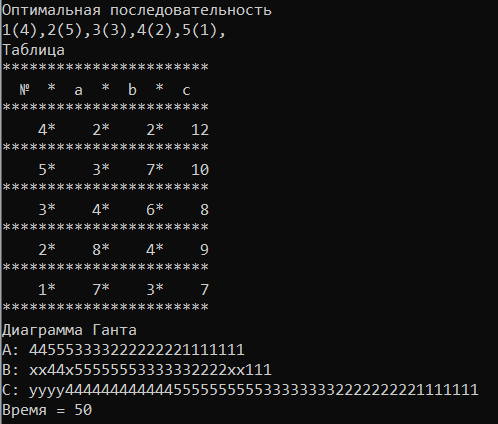
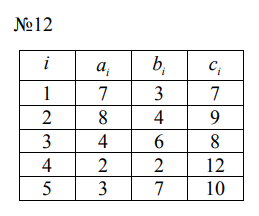


Рис 2. Таблица оптимальных значений, оптимальная последовательность, диаграмма Ганта, оптимальное время работы.

# **Задание 3. Задача упорядочивания nx3 решаемая методом перебора**

Не выполнение условия Джонсона не дает возможности свести задачу упорядочения nx3 к задаче упорядочения nx2. Для поиска оптимального решения остается лишь метод перебора, который будет гарантировать, что найденное решение будет самым оптимальным. Общее количество вариантов решения задачи равно n!, где n – количество деталей.

**Вариант 12:**



## Решение задачи задания 3

### **Алгоритм решения:**

Оптимизация происходит в функции Perebor, которая запускается, если не выполнено условие из 2 задания (выделенный сегмент в функции NX3). Сначала происходит вычисление текущего времени работы и данное время запоминается в переменную. Исходная таблица также запоминается. Затем происходит цикл по перестановкам на каждой итерации которого находится новая перестановка и проверяется значение времени. Если новое время работы меньше предыдущего, то новая перестановка и время запоминаются в соответствующие переменные. В конце цикла получим таблицу с оптимальным временем работы. Поиск перестановок происходит по алгоритму:

1. Необходимо просмотреть текущую перестановку справа налево и при этом следить за тем, чтобы каждый следующий элемент перестановки (элемент с большим номером) был не более чем предыдущий (элемент с меньшим номером). Как только данное соотношение будет нарушено необходимо остановиться и отметить текущее число (позиция 1).
2. Снова просмотреть пройденный путь справа налево пока не дойдем до первого числа, которое больше чем отмеченное на предыдущем шаге.
3. Поменять местами два полученных элемента.
4. Теперь в части массива, которая размещена справа от позиции 1 надо отсортировать все числа в порядке возрастания. Поскольку до этого они все были уже записаны в порядке убывания необходимо эту часть подпоследовательность просто перевернуть.

## Код программы на языке С#

Фрагмент, относящийся к Заданию №3.

### **Код метода перестановок и вычисления оптимальной перестановки:**

### public static void Perebor(List<Tabel> tab)

### {

### int j = tab.Count - 2;

### int k, l, r;

### int midT, minT = -1;

### List<Tabel> minTab = new List<Tabel>(tab);

### minT = FindTime(tab);

### while(j != -1)

### {

### while (j != -1 && tab[j].num >= tab[j + 1].num) j--;

### if (j != -1)

### {

### k = tab.Count - 1;

### while (tab[j].num >= tab[k].num) k--;

### Swap(tab, j, k);

### l = j + 1;

### r = tab.Count - 1;

### while (l < r)

### {

### Swap(tab, l, r);

### l++;

### r--;

### }

### midT = FindTime(tab);

### if (midT < minT) minTab = new List<Tabel>(tab);

### }

### }

### tab = minTab;

### }

### **Метод нахождения времени работы:**

### public static int FindTime(List<Tabel> tab)

### {

### int Ta, Tb, Tc;

### Ta = Tb = Tc = 0;

### for (int i = 0; i < tab.Count; i++)

### {

### Ta += tab[i].a;

### if (Ta - Tb > 0)

### {

### Tb += Ta - Tb;

### }

### Tb += tab[i].b;

### if (Tb - Tc > 0)

### {

### Tc += Tb - Tc;

### }

### Tc += tab[i].c;

### }

### return Tc;

### }

### **Код Main:**

## static void Main(string[] args)

## {

## ……….

## //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## //3 задание

## //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Console.WriteLine("3 задание");

## Console.WriteLine("Исходная таблица");

## ShowTabel(PereborTabel, 2);

## Console.WriteLine("Диаграмма Ганта");

## help = Gant(PereborTabel, 2);

## Console.WriteLine("Время = " + help);

## Perebor(PereborTabel);

## Console.WriteLine("Оптимальная последовательность");

## ShowWay(PereborTabel);

## Console.WriteLine("Таблица");

## ShowTabel(PereborTabel, 2);

## Console.WriteLine("Диаграмма Ганта");

## help = Gant(PereborTabel, 2);

## Console.WriteLine("Время = " + help);

## Console.ReadLine();

## }

## Результаты выполнения задания №3

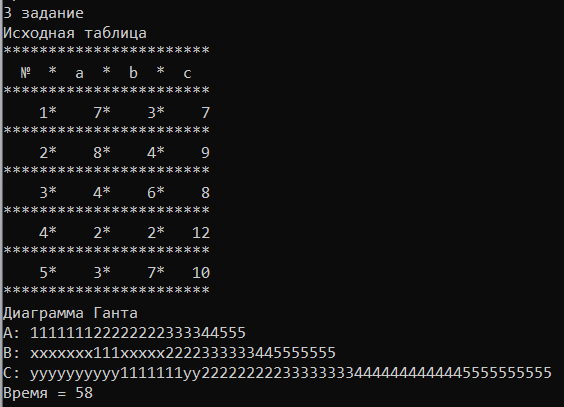


Рис 1. Исходная таблица значений, исходный маршрут, диаграмма Ганта, текущее время работы и проверка условия сводимости

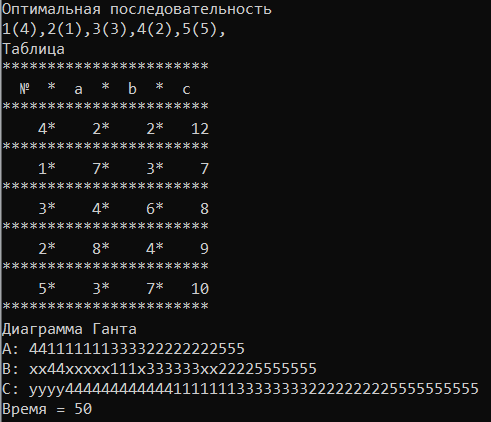


Рис 2. Таблица оптимальных значений, оптимальная последовательность, диаграмма Ганта, оптимальное время работы.

# 

# **Заключение**

В данной лабораторной были рассмотрены задачи упорядочивания nx2 и nx3 из теории расписаний, были изучены соответствующие модели и алгоритмы нахождения оптимального решения. При выполнении лабораторной работы были изучены методы полного перебора для нахождения оптимального решения. Кроме того был реализован вывод последовательности работ в виде диаграммы Ганта для таблиц nx2 и nx3.

# 

# **Список используемых источников**

1. Сорокин А.В. Использование алгоритма Джонсона для решения задачи упорядочения. Методические указания. - Барнаул: АлтГТУ, 2015. – 32 с. – [Электронный ресурс]. – url: http://elib.altstu.ru/eum/download/pm/Sorokin\_alg\_Johnson.pdf
2. Алпатов, Ю.Н. Математическое моделирование производственных процессов [Электронный ресурс] : учеб. пособие / Ю.Н. Алпатов. — Электрон. дан. — Санкт-Петербург : Лань, 2018. — 136 с. — Режим доступа: https://e.lanbook.com/book/107271. — Загл. с экрана.