|  |  |  |
| --- | --- | --- |
|  | gerb |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Кибернетики

Кафедра информационной безопасности

|  |  |
| --- | --- |
| **КУРСОВАЯРАБОТА** | |
| **по дисциплине** | |
| **«**Математическая логика**»** | |
| **Тема курсовой работы:**  **«Методы разработки алгоритмов»** | |
| Студент группы \_\_\_\_ККСО-04-19\_\_\_\_ | *Савилов Д.А.* |
| Руководитель курсовой работы | *Кузнецов В.С.* |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2022 г. | *(подпись студента)* |
|  |  |  |
| Допущен к защите | «\_\_»\_\_\_\_\_\_\_2022 г. | *(подпись руководителя)* |

Москва, 2022

Оглавление

[**Введение 3**](#_Toc90373609)

[**Основная часть 4**](#_Toc90373610)

[**1. Метод грубой силы (Brute Force) 4**](#_Toc90373611)

[**2. Метод частных целей 4**](#_Toc90373612)

[**3. Метод «Разделяй и властвуй». 10**](#_Toc90373613)

[**4. Жадный алгоритм 11**](#_Toc90373614)

[**5. Эвристические методы 12**](#_Toc90373615)

[**6. Метод обрабатывания назад 13**](#_Toc90373616)

[**7. Метод рекуррентных соотношений 15**](#_Toc90373617)

[**Заключение 17**](#_Toc90373618)

[**Список литературы 19**](#_Toc90373619)

## Введение

На данный момент нет универсального способа, позволяющего без особого труда составлять любые алгоритмы. К сожалению, такого способа не существует, ведь жизненные ситуации и задачи так разнообразны и непредсказуемы.

Однако выбор конкретного численного метода решения задачи обычно производится по следующим критериям:

1. обеспечение оптимального времени решения задачи;
2. обеспечение оптимального использования имеющихся ресурсов (памяти);
3. обеспечение требуемой точности вычислений;
4. минимальные стоимостные затраты
5. возможность использования стандартных подпрограмм.

При дальнейшей постановке задачи на ПК отыскивается наиболее рациональный способ решения задачи.

Разработка алгоритма, являясь четким логичным процессом, упрощается на каждом уровне шаг за шагом. Затем в процессе задействуется следующий метод алгоритмизации - метод пошагового уточнения (совершенствования). Сначала задача рассматривается в целом, выделяются наиболее крупные ее части. Алгоритм, указывающий порядок выполнения этих частей, описывается в структурированной форме, не вдаваясь в мелкие детали. Затем от общей структуры переходят к описанию отдельных частей. Таким образом, разработка алгоритма состоит из последовательности шагов в направлении уточнения алгоритма.

К настоящему времени разработано множество алгоритмических стратегий, определены ключевые принципы, лежащие в основе этих стратегий, что позволяет их применять как универсальный инструментарий для широкого класса задач.

Создано множество алгоритмов, которые могут служить фундаментом современного компьютерного программирования и которые можно использовать как строительные блоки при программировании.

Очевидно, что для решения одной и той же задачи можно создавать разные алгоритмы в зависимости от выбора алгоритмической стратегии.

Для того чтобы успешно использовать алгоритмы, созданные в соответствии с выбранной стратегией, недостаточно просто подготовить код программы.

Необходимо знать, как различные алгоритмы ведут себя в разных ситуациях. В конечном итоге именно эта информация определяет выбор стратегии, а также позволяет критически оценивать новые алгоритмы.

## Основная часть

Рассмотрим несколько методик построения различных алгоритмов:

### Метод грубой силы (Brute Force)

Метод грубой силы (brute force) – решение “в лоб”, основан на прямом подходе к решению задачи и опирается на определения понятий, используемых в постановке задачи

**Пример: Задача возведения числа 𝑎 в неотрицательную степень 𝑛**

Алгоритм решения “в лоб”

По определению

Отсюда имеем простейший алгоритм:

***f****unction pow(a, n)*

*pow = a*

*for i = 2 to n do*

*pow = pow \* a*

*end for*

*return pow*

*end function*

### **Метод частных целей**

Метод частных целей состоит в том, что первоначальная задача сводится к последовательности более простых задач. Поскольку более простые задачи легче обрабатывать по сравнению с первоначальной задачей, то ее решение может быть получено из решений более простых задач. При этом не существует общего набора правил как для выбора более простых задач, так и для определения класса задач, которые можно решить с помощью данного подхода.

Для создания алгоритма методом частных целей необходимо ответить

на следующие вопросы:

1. Можно ли решить задачу, не учитывая некоторые условия?
2. Можно ли решить задачу для некоторых частных случаев?
3. Можно ли упростить задачу, наложив дополнительные ограничения?
4. Можно ли разработать алгоритм, который дает решение при всех условиях задачи, но при ограничениях на входные данные?
5. Существуют ли решения аналогичных задач и нельзя ли эти решения

модифицировать для решения поставленной задачи?

Достоинство метода:

· естественный (и в чем-то интуитивный) подход к решению сложной проблемы.

Недостатки метода:

· осмысленный выбор более простых задач не имеет научного обоснования и во многом зависит от опыта и интуиции специалиста;

· отсутствуют правила для определения задач, которые могут быть решены таким методом.

**Пример**:   
В качестве примера возьмем задачу сетевого планирования «Метод критического пути», представленную в [1]:

Построить сетевой график, рассчитать наиболее ранние и наиболее поздние сроки наступления событий, найти критический путь, определить полные и независимые резервы времени всех работ и коэффициенты напряженности некритических дуг с помощью данных, представленных в след. виде:

Сначала строим структурный сетевой график и вводим правильную нумерацию событий

Изображение выглядит как другой, провод, упорядочено

Автоматически созданное описание

Рис. 1 Сетевой график

Наиболее ранние сроки наступления событий находим по формуле

где максимум берется по всем событиям j, непосредственно предшествующим событию i. Начальному событию присваиваем Тогда:

Итак, критическое время . Минимальный срок выполнения проекта – 19 дней.

Наиболее поздние сроки наступления событий находим по формуле

где минимум берется по всем событиям j, непосредственно следующим за событием i. Конечному событию присваиваем наиболее поздний срок наступления, равный критическому времени:

Тогда:

Результаты расчетов отразим на сетевом графике. Ранние сроки наступления событий запишем над кружками, изображающими эти события, поздние сроки наступления событий – под кружками

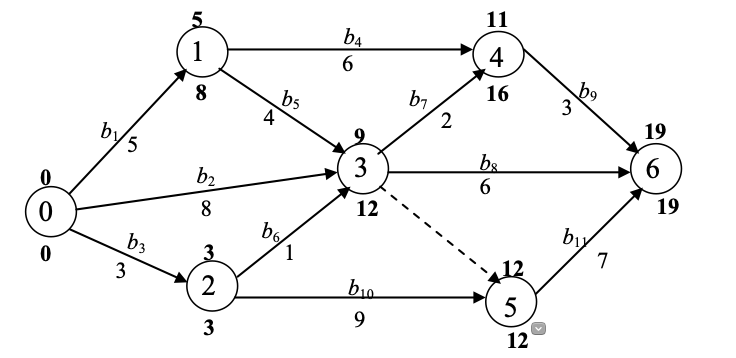


Рис. 2 Сетевой график с расставленными событиями и временем наступления

Критическое время   
Временные характеристики событий представлены в таблице: Изображение выглядит как стол

Автоматически созданное описание

Резервы времени событий найдены по формуле Критический путь проходит через события с нулевым резервом времени, т. е. через события 0, 2, 5, 6.

Найдем резервы времени работ.

Наиболее ранний возможный срок начала работы равен наиболее раннему сроку наступления события i а наиболее поздний допустимый срок окончания работы равен наиболее позднему сроку наступления события j:   
Полный резерв времени работ найдем по формуле

Независимый резерв времени работ найдем по формуле

.

Сведем полученные данные в таблицу:

Изображение выглядит как стол

Автоматически созданное описание

Работа – фиктивная работа.

Критические работы – . Резервы времени этих работ равны нулю. Выделим критический путь двойными стрелками .

Изображение выглядит как часы, лампа

Автоматически созданное описание

Рис. 3 Критический путь

Резерв времени некритической дуги b находим как разность между длиной замыкающего критического участка и длиной самой некритической дуги:

Коэффициент напряженности некритической дуги определим по формуле

Резервы времени и коэффициенты напряженности некритических дуг: Изображение выглядит как стол

Автоматически созданное описание

Дуги, коэффициент напряженности которых составляют критическую зону, дуги с коэффициентом напряженности образуют подкритическую зону, а дуги с коэффициентом дают резервную зону. В нашем случае в критическую зону попадает только критический путь, в подкритической зоне находятся дуги (0, 1, 3, 6), (0, 1, 3, 5), (0,3, 6), (0, 1, 4, 6), (0, 1, 3, 4, 6) и (0, 3, 5). Из них самая напряженная дуга (0, 1, 3, 6). Она быстрее других может перейти на критический путь. Дуги (2, 3, 5), (2, 3, 6) и (2, 3, 4, 6) образуют резервную зону.

### Метод «Разделяй и властвуй».

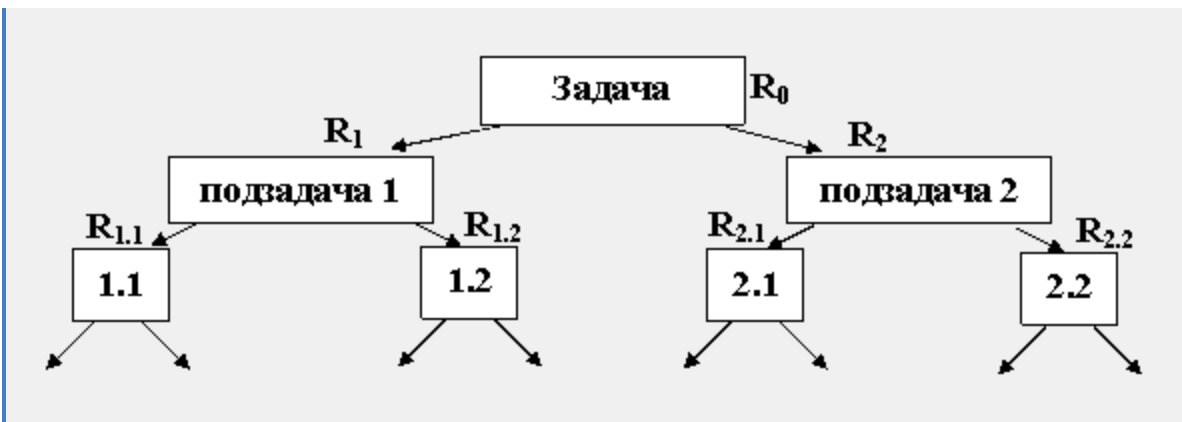


Рис. 4 Структура метода

При решении задачи необходимо ответить на следующие вопросы:

1.  Известно ли решение для какой-либо части задачи и можно ли решить оставшуюся неизвестной часть.

2.  Известны ли частные случаи решения задачи, и можно ли разработать алгоритм решения задачи для ограниченного подмножества исходных операндов.

3.  Существует ли в задаче неясные моменты (раскрытие глубины задачи).

4.  Существует ил похожая задача того же класса и решение для нее. Можно ли изменить алгоритм решения второй задачи для получения решения первой.

В основе метода «Разделяй и властвуй» лежит разбиение задачи на подзадачи (декомпозиция).

Первый шаг – это разделение задачи на К подзадач со сложностью 1/К.

Властвование – второй шаг алгоритма. Это – использование рекурсионного процесса разбиения подзадач на еще более мелкие подзадачи до тех пор, пока подзадачи n-го уровня разделения будут достаточно малы для их тривиального решения.

Третий этап метода – склеивание множества решений отдельных подзадач в общее решение.

**Пример: Сортировка слиянием**

1. Массив рекурсивно разбивается пополам, и каждая из половин делиться до тех пор, пока размер очередного подмассива не станет равным единице;
2. Далее выполняется операция алгоритма, называемая слиянием. Два единичных массива сливаются в общий результирующий массив, при этом из каждого выбирается меньший элемент (сортировка по возрастанию) и записывается в свободную левую ячейку результирующего массива.
3. После чего из двух результирующих массивов собирается третий общий отсортированный массив, и так далее. В случае если один из массивов закончиться, элементы другого дописываются в собираемый массив;

В конце операции слияния, элементы перезаписываются из результирующего массива в исходный.

### Жадный алгоритм

«Жадный алгоритм» позволяет получить оптимальный результат в целом. Однако в жизни «жадная» стратегия дает стоимостную выгоду, тогда как общий результат может оказаться неприемлемым.

**Пример: Алгоритм Крускала (нахождение минимального остовного дерева)**

Начинаем с ребер с наименьшим весом и продолжаем добавлять ребра, пока не достигнем нашей цели.

Шаги для реализации алгоритма Крускала следующие:

1. Сортировать все ребра от малого веса до высокого.
2. Возьмите ребро с наименьшим весом и добавьте его в остовное дерево. Если добавление ребра создало цикл, то отклоните это ребро.
3. Продолжайте добавлять ребра, пока не достигнете всех вершин.

Дан граф

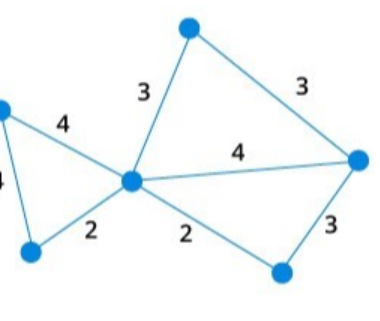


Рис. 5 Исходный граф

1. Выбрать ребро с наименьшим весом (больше 1 – выбрать любое)
2. Выбрать следующее наикратчайшее ребро и добавить его
3. Выбрать следующее наикратчайшее ребро, не образующее цикл, и добавить его
4. Повторять до тех пор, пока не получим остовное дерево
5. Итого, получим

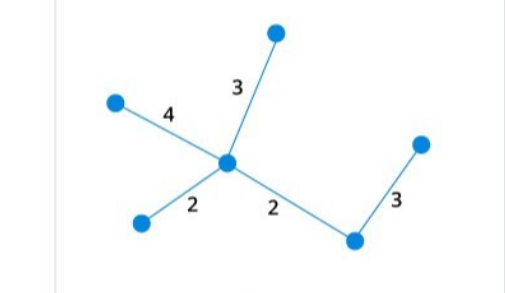


Рис 6. Минимальное остовное дерево

### 5. Эвристические методы

Под эвристическими понимаются методы, правильность  
которых не доказана. Они выглядят правдоподобными, кажется, что в большинстве случаев они должны давать верное решение. Иногда не удается построить контрпример, демонстрирующий ошибочность или неуниверсальность метода. Но не удается доказать математическими средствами и правильность метода. Тем не менее, практика использования эвристических методов дает положительные результаты.

Эвристические методы разнообразны, поэтому нельзя описать какую-то общую схему разработки таких методов. Чаще всего эвристические методы применяют совместно с методами перебора для сокращения количества проверяемых вариантов: некоторые подмножества вариантов согласно выбранной эвристике считаются заведомо неприемлемыми и не проверяются. Таким образом, переборный алгоритм с эвристикой выполняется гораздо быстрее, чем алгоритм полного перебора. Платой за это является отсутствие гарантии правильности решения или гарантии того, что из всех возможных выбрано наилучшее решение.

**Пример: задача раскраски вершин графа**

Раскрасить вершины графа таким образом, чтобы смежные вершины были раскрашены в различные цвета, а количество использованных в графе красок было минимальным.

Точное решение задачи возможно методом полного перебора, но оно будет получаться за слишком большое время для графов, содержащих несколько десятков вершин. Однако можно заметить, что в формулировке задачи содержатся два условия - первое - обязательное и второе (минимальности), которое зачастую можно ослабить. Например, если минимальное число красок равно десяти, а алгоритм быстро найдет раскраску,  
использующую одиннадцать красок, то часто такое решение можно рассматривать как приемлемое.

### Метод обрабатывания назад

Решение задачи в этом случае должно быть известно заранее. От него начинается обратное движение к ее начальной постановке. Если эти действия обратимы, то можно организовать обратный процесс от постановки задачи к ее решению.

**Пример: Задача нахождения кратчайшего маршрута методом динамического программирования**

Рассмотрим решение согласно [3].

Вводится функция , определяющая минимальную длину из начальной вершины в вершину i. Через обозначают длину пути между вершинами i и j, а – наименьшую длину пути между вершиной j и начальной вершиной.

Выбирая в качестве i такую вершину, которая минимизирует сумму  , получают функциональное уравнение Беллмана

Сложность решения данного уравнения существенно зависит от вида графа, описывающего транспортную сеть. Если граф направленный, задача решается достаточно просто.

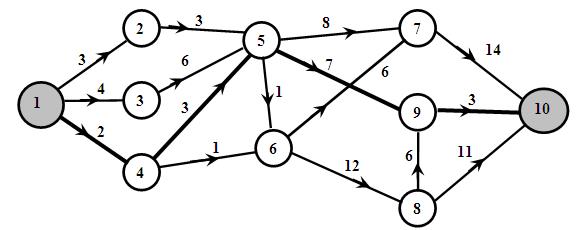


Рис. 7 Исходный граф

Для упрощения и систематизации составления функций Беллмана рационально пронумеровать вершина так, чтобы дуга выходила из вершины с меньшим номером. Граф на Рис.1 этому требованию удовлетворяет. В этом случае последовательно находят функции fi для каждой вершины ориентированного графа.



Длина кратчайшего пути составляет 15 условных единиц. Для выбора оптимальной траектории осуществляют просмотр функций fi в обратном порядке. В данном случае:

Минимальная выбранная сумма 3+12=15 соответствует вершине с номером 9.

При вычислении f9 была выбрана вершина 5. Продолжая таким же образом, получим кратчайший путь от вершины 1 к вершине 10 (1, 4, 5, 9, 10)

На рисунке дуги этого пути изображены жирными линиями.

### Метод рекуррентных соотношений

Идея метода рекуррентных соотношений чрезвычайно проста — на основе некоторых рассуждений, строится рекуррентное соотношение, решение которого доставляет решение нашей задачи, и на основе этого рекуррентного соотношения разрабатывается рекурсивный алгоритм. Отметим, что фраза «используя некоторые рассуждения» совершенно не определяет конкретный метод получения рекуррентного соотношения для определенной задачи. Мы можем лишь уточнить, что эти рассуждения должны отражать наше рекурсивное понимание структуры задачи, т. е. следовать схеме понижения аргумента или размерности. Решение для некоторой размерности задачи или для некоторого аргумента функции должно быть сформулировано на основе ее сведения к задачам меньшей размерности или функциям с меньшим значением аргумента. Условие останова рекурсии позволяет решить задачу при некоторых малых размерностях или вычислить функцию при начальных значениях аргумента. Приведем примеры использования этого метода с комментариями к этапам разработки рекуррентного соотношения

**Пример: Нахождение предыдущего и последующего числа, опираясь на аксиомы Пеано.**

Приведем формулировку одной их аксиом Пеано «Для каждого натурального числа x имеется, и притом только одно, натуральное число, называемое его последующим и обозначаемое x′ ».

В целях удобства записи будем обозначать операцию, задающую для целого неотрицательного числа n его последующее число через n +1, а операцию, задающую для целого положительного числа n его предыдущее число через n −1. Наша задача состоит в том, что бы разработать рекурсивный алгоритм для вычисления суммы двух чисел n, m в этой модели вычислений. Условие останова рекурсии легко формулируется: n + m = n, m = 0 . Выберем число m в качестве аргумента рекурсии — нам необходимо записать сумму при уменьшении аргумента на единицу с использованием разрешенных в нашей модели операций. Решение достаточно очевидно:

n + m = (n +1) + (m −1)

Проведенные рассуждения позволяют записать рекуррентное соотношение, для рекурсивно заданной функции S(n,m), значением которой является сумма ее аргументов:

## Заключение

Рассмотренные в данной работе методы разработки алгоритмов играют огромную роль в повседневной жизни. Наш мир оказался под влиянием знаний, порождающих развитие все новых современных технологий, разработку продуктов, создание которых ранее считалось невозможным, интеллектуальное развитие специалистов, воплощающих многочисленные идеи в жизнь. Для каждой задачи может существовать множество алгоритмов, приводящих к цели и выбор правильного метода их разработки напрямую влияет на увеличение эффективности алгоритмов, что является одной из главных задач современной информатики.

## Список литературы

1. Задачи сетевого планирования [Электронный ресурс]: учебное пособие/М. А. Плескунов
2. Алгоритм Крускала [Электронный ресурс]: <https://evileg.com/ru/users/mafulechka/albums/photo/825/>
3. Выбор оптимального маршрута методом динамического программирования [Электронный ресурс]:<https://helpiks.org/2-66840.html#:~:text=Задача%20нахождения%20кратчайшего%20маршрута%20методом,вершиной%20j%20и%20начальной%20вершиной>
4. Теория рекурсии для программистов [Электронный ресурс]: учебное пособие/В. А. Головешкин, М. В. Ульянов
5. Теория алгоритмов [Электронный ресурс]: Учебное пособие/ Алябьева В.Г., Пастухова Г.В
6. Разработка алгоритмов для решения задач на эвм [Электронный ресурс]: Учебное пособие/ В.И. Перова, Т.А. Сабаева, Д.Т. Чекмарев