# Лекции

# по предмету «Экспертные информационные системы»

# группы СЗБ-189с, 289с

План:

1. Введение в экспертные информационные системы
2. Общие сведения о графах:
    - 2.1. Определение графа
    - 2.2. Матрица смежности и матрица инцидентности
    - 2.3. Цикломатическое число
    - 2.4. Хроматическое число
    - 2.5. Метрические характеристики графа
    - 2.6. Алгорим Декстры

3. Транспортные задачи

## 1. Введение в экспертные информационные системы

В начале восьмидесятых годов в исследованиях по искусственному интеллекту сформировалось самостоятельное направление, получившее название "экспертные системы" (ЭС). Цель исследований по ЭС состоит в разработке программ, которые при решении задач, трудных для эксперта-человека, получают результаты, не уступающие по качеству и эффективности решениям, получаемым экспертом. Исследователи в области ЭС для названия своей дисциплины часто используют также термин "инженерия знаний", введенный Е.Фейгенбаумом как "привнесение принципов и инструментария исследований из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов".

Программные средства (ПС), базирующиеся на технологии экспертных систем, или инженерии знаний (в дальнейшем будем использовать их как синонимы), получили значительное распространение в мире. Важность экспертных систем состоит в следующем:

технология экспертных систем существенно расширяет круг практически значимых задач, решаемых на компьютерах, решение которых приносит значительный экономический эффект;

технология ЭС является важнейшим средством в решении глобальных проблем традиционного программирования: длительность и, следовательно, высокая стоимость разработки сложных приложений;

высокая стоимость сопровождения сложных систем, которая часто в несколько раз превосходит стоимость их разработки; низкий уровень повторной используемости программ и т.п.;

объединение технологии ЭС с технологией традиционного программирования добавляет новые качества к программным продуктам за счет: обеспечения динамичной модификации приложений пользователем, а не программистом; большей "прозрачности" приложения (например, знания

хранятся на ограниченном ЕЯ, что не требует комментариев к знаниям, упрощает обучение и сопровождение); лучшей графики; интерфейса и взаимодействия.

По мнению ведущих специалистов , в недалекой перспективе ЭС найдут следующее применение:

ЭС будут играть ведущую роль во всех фазах проектирования, разработки, производства, распределения, продажи, поддержки и оказания услуг;

технология ЭС, получившая коммерческое распространение, обеспечит революционный прорыв в интеграции приложений из готовых интеллектуально-взаимодействующих модулей.

ЭС предназначены для так называемых неформализованных задач, т.е. ЭС не отвергают и не заменяют традиционного подхода к разработке программ, ориентированного на решение формализованных задач.

Неформализованные задачи обычно обладают следующими особенностями:

ошибочностью, неоднозначностью, неполнотой и противоречивостью исходных данных;

ошибочностью, неоднозначностью, неполнотой и противоречивостью знаний о проблемной области и решаемой задаче;

большой размерностью пространства решения, т.е. перебор при поиске решения весьма велик;

динамически изменяющимися данными и знаниями.

Следует подчеркнуть, что неформализованные задачи представляют большой и очень важный класс задач. Многие специалисты считают, что эти задачи являются наиболее массовым классом задач, решаемых ЭВМ.

Экспертные системы и системы искусственного интеллекта отличаются от систем обработки данных тем, что в них в основном используются символьный (а не числовой) способ представления, символьный вывод и эвристический поиск решения (а не исполнение известного алгоритма).

Экспертные системы применяются для решения только трудных практических (не игрушечных) задач. По качеству и эффективности решения экспертные системы не уступают решениям эксперта-человека. Решения экспертных систем обладают *"прозрачностью",* т.е. могут быть объяснены пользователю на качественном уровне. Это качество экспертных систем обеспечивается их способностью рассуждать о своих знаниях и умозаключениях. Экспертные системы способны пополнять свои знания в ходе взаимодействия с экспертом. Необходимо отметить, что в настоящее время технология экспертных систем используется для решения различных типов задач (интерпретация, предсказание, диагностика, планирование, конструирование, контроль, отладка, инструктаж, управление ) в самых разнообразных проблемных областях, таких, как финансы, нефтяная и газовая промышленность, энергетика, транспорт, фармацевтическое производство, космос, металлургия, горное дело, химия, образование, целлюлозно-бумажная промышленность, телекоммуникации и связь и др.

Коммерческие успехи к фирмам-разработчикам систем искусственного интеллекта (СИИ) пришли не сразу. На протяжении 1960 - 1985 гг. успехи ИИ касались в основном исследовательских разработок, которые демонстрировали пригодность СИИ для практического использования. Начиная примерно с 1985 г. (в массовом масштабе с 1988 - 1990 гг.), в первую очередь ЭС, а в последние годы системы, воспринимающие естественный язык (ЕЯ-системы), и нейронные сети (НС) стали активно использоваться в коммерческих приложениях.

Следует обратить внимание на то, что некоторые специалисты (как правило, специалисты в программировании, а не в ИИ) продолжают утверждать, что ЭС и СИИ не оправдали возлагавшихся на них ожиданий и умерли. Причины таких заблуждений состоят в том, что эти авторы рассматривали ЭС как альтернативу традиционному программированию, т.е. они исходили из того, что ЭС в одиночестве (в изоляции от других программных средств) полностью решают задачи, стоящие перед заказчиком.

Надо отметить, что на заре появления ЭС специфика используемых в них языков, технологии разработки приложений и используемого оборудования (например, Lisp-машины) давала основания предполагать, что интеграция ЭС с традиционными, программными системами является сложной и, возможно, невыполнимой задачей при ограничениях, накладываемых реальными приложениями. Однако в настоящее время коммерческие инструментальные средства (ИС) для создания ЭС разрабатываются в полном соответствии с современными технологическими тенденциями традиционного программирования, что снимает проблемы, возникающие при создании интегрированных приложений.

Причины, приведшие СИИ к коммерческому успеху, следующие.

*Интегрированность*. Разработаны инструментальные средства искусственного интеллекта (ИС ИИ), легко интегрирующиеся с другими информационными технологиями и средствами (с CASE, СУБД, контроллерами, концентраторами данных и т.п.).

*Открытость и переносимость*. ИС ИИ разрабатываются с соблюдением стандартов, обеспечивающих открытость и переносимость [14].

*Использование языков традиционного программирования и рабочих станций*. Переход от ИС ИИ, реализованных на языках ИИ (Lisp, Prolog и т.п.), к ИС ИИ, реализованным на языках традиционного программирования (C, C++ и т.п.), упростил обеспечение интегриро-ванности, снизил требования приложений ИИ к быстродействию ЭВМ и объемам оперативной памяти. Использование рабочих станций (вместо ПК) резко увеличило круг приложений, которые могут быть выполнены на ЭВМ с использованием ИС ИИ.

*Архитектура клиент-сервер*. Разработаны ИС ИИ, поддерживающие распределенные вычисления по архитектуре клиент-сервер, что позволило:снизить стоимость оборудования, используемого в приложениях, децентрализовать приложения, повысить надежность и общую производительность (так как сокращается количество информации,

пересылаемой между ЭВМ, и каждый модуль приложения выполняется на адекватном ему оборудовании).

*Проблемно/предметно-ориентированные ИС ИИ.* Переход от разработок ИС ИИ общего назначения (хотя они не утратили свое значение как средство для создания ориентированных ИС) к проблемно/предметно-ориентированным ИС ИИ [9] обеспечивает: сокращение сроков разработки приложений; увеличение эффективности использования ИС; упрощение и ускорение работы эксперта; повторную используемость информационного и программного обеспечения (объекты,классы,правила,процедуры).

Структура экспертных систем

Типичная статическая ЭС состоит из следующих основных компонентов (рис. 1.):

- решателя (интерпретатора);
- рабочей памяти (РП), называемой также базой данных (БД);
- базы знаний (БЗ);
- компонентов приобретения знаний;
- объяснительного компонента;
- диалогового компонента.

*База данных (рабочая память)* предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи. Этот термин совпадает по названию, но не по смыслу с термином, используемым в информационно-поисковых системах (ИПС) и системах управления базами данных (СУБД) для обозначения всех данных (в первую очередь долгосрочных), хранимых в системе.

*База знаний* (БЗ) в ЭС предназначена для хранения долгосрочных данных, описывающих рассматриваемую область (а не текущих данных), и правил, описывающих целесообразные преобразования данных этой области.

*Решатель,* используя исходные данные из рабочей памяти и знания из БЗ, формирует такую последовательность правил, которые, будучи примененными к исходным данным, приводят к решению задачи.

*Компонент* приобретения знаний автоматизирует процесс наполнения ЭС знаниями, осуществляемый пользователем-экспертом.

*Объяснительный компонент* объясняет, как система получила решение задачи (или почему она не получила решение) и какие знания она при этом использовала, что облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату.

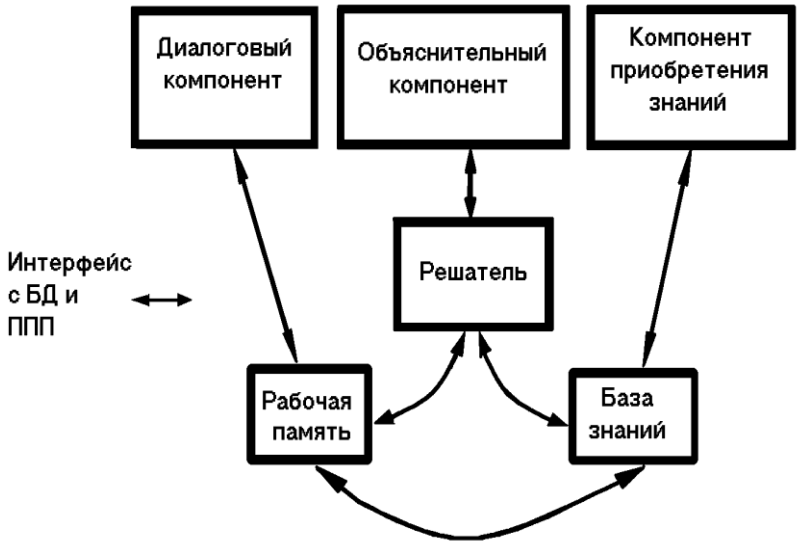



Рис.1. Структура статической ЭС.

*Диалоговый компонент* ориентирован на организацию дружественного общения с пользователем как в ходе решения задач, так и в процессе приобретения знаний и объяснения результатов работы.

В разработке ЭС участвуют представители следующих специальностей:

эксперт в проблемной области, задачи которой будет решать ЭС;

инженер по знаниям - специалист по разработке ЭС (используемые им технологию, методы называют технологией (методами) инженерии знаний);

программист по разработке инструментальных средств (ИС), предназначенных для ускорения разработки ЭС.

Необходимо отметить, что отсутствие среди участников разработки инженеров по знаниям (т. е. их замена программистами) либо приводит к неудаче процесс создания ЭС, либо значительно удлиняет его.

*Эксперт* определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

*Инженер по знаниям* помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того ИС, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний в этом ИС; выделяет и программирует (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом.

*Программист* разрабатывает ИС (если ИС разрабатывается заново), содержащее в пределе все основные компоненты ЭС, и осуществляет его сопряжение с той средой, в которой оно будет использовано.

Экспертная система работает в двух режимах: режиме приобретения знаний и в режиме решения задачи (называемом также режимом консультации или режимом использования ЭС).

*В режиме приобретения знаний* общение с ЭС осуществляет (через посредничество инженера по знаниям) эксперт. В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерные для рассматриваемой области.

Отметим, что режиму приобретения знаний в традиционном подходе к разработке программ соответствуют этапы алгоритмизации,

программирования и отладки, выполняемые программистом. Таким образом, в отличие от традиционного подхода в случае ЭС разработку программ осуществляет не программист, а эксперт (с помощью ЭС), не владеющий программированием.

*В режиме консультации* общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения. Необходимо отметить, что в зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (в этом случае он обращается к ЭС за результатом, не умея получить его сам), или быть специалистом (в этом случае пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, либо возложить на ЭС рутинную работу). В режиме консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память. Решатель на основе входных данных из рабочей памяти, общих данных о проблемной области и правил из БЗ формирует решение задачи. ЭС при решении задачи не только исполняет предписанную последовательность операции, но и предварительно формирует ее. Если реакция системы не понятна пользователю, то он может потребовать объяснения:

"Почему система задает тот или иной вопрос?", "как ответ, собираемый системой, получен?".

Структуру, приведенную на рис. 1.1, *называют структурой статической ЭС.* ЭС данного типа используются в тех приложениях, где можно не учитывать изменения окружающего мира, происходящие за время решения задачи. Первые ЭС, получившие практическое использование, были статическими.

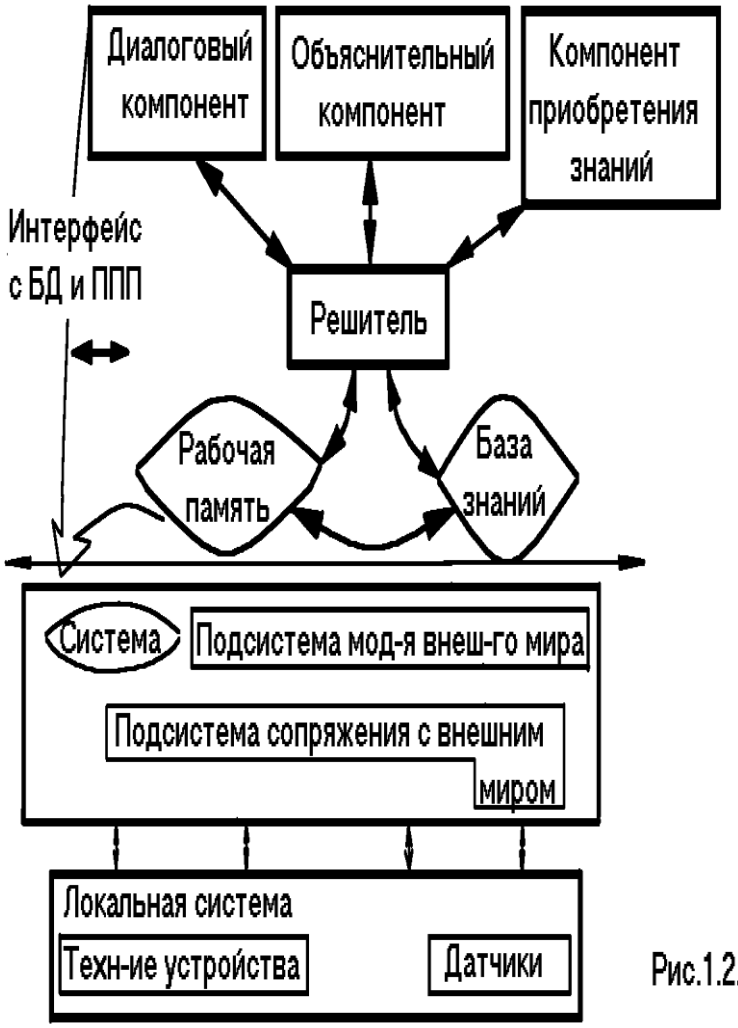



Рис.1.2.

На рис. 1.2 показано, что в архитектуру динамической ЭС по сравнению со статической ЭС вводятся два компонента: подсистема моделирования внешнего мира и подсистема связи с внешним окружением. Последняя осуществляет связи с внешним миром через систему датчиков и контроллеров. Кроме того, традиционные компоненты статической ЭС (база знаний и машина вывода) претерпевают существенные изменения, чтобы отразить временную логику происходящих в реальном мире событий.

Подчеркнем, что структура ЭС, представленная на рис. 1.1 и 1.2, отражает только компоненты (функции), и многое остается "за кадром". На рис. 1.3 приведена обобщенная структура современного ИС для создания динамических ЭС, содержащая кроме основных компонентов те возможности, которые позволяют создавать интегрированные приложение в соответствии с современной технологией программирования.

**Этапы разработки экспертных систем**

Разработка ЭС имеет существенные отличия от разработки обычного программного продукта. Опыт создания ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату.

Использовать ЭС следует только тогда, когда разработка ЭС *возможна, оправдана и* методы инженерии знаний *соответствуют* решаемой задаче. Чтобы разработка ЭС была *возможной* для данного приложения, необходимо одновременное выполнение по крайней мере следующих требований:

1) существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;

2) эксперты сходятся в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;

3) эксперты способны вербализовать (выразить на естественном языке) и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что знания экспертов будут "извлечены" и вложены в ЭС;

4) решение задачи требует только рассуждений, а не действий;

5) задача не должна быть слишком трудной (т.е. ее решение должно занимать у эксперта несколько часов или дней, а не недель);

6) задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно "понятной" и структурированной области, т.е. должны быть выделены основные понятия, отношения и известные (хотя бы эксперту) способы получения решения задачи;

7) решение задачи не должно в значительной степени использовать "здравый смысл" (т.е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный человек), так как подобные знания пока не удается (в достаточном количестве) вложить в системы искусственного интеллекта.

Использование ЭС в данном приложении может быть возможно, но не оправдано. Применение ЭС может быть *оправдано* одним из следующих факторов:

решение задачи принесет значительный эффект, например экономический;

использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;

использование ЭС целесообразно в тех случаях, когда при передаче информации эксперту происходит недопустимая потеря времени или информации;

использование ЭС целесообразно при необходимости решать задачу в окружении, враждебном для человека.

Приложение *соответствует* методам ЭС, если решаемая задача обладает совокупностью следующих характеристик:

1) задача может быть естественным образом решена посредством манипуляции с символами (т.е. с помощью символических рассуждений), а не манипуляций с числами, как принято в математических методах и в традиционном программировании;

2) задача должна иметь эвристическую, а не алгоритмическую природу, т.е. ее решение должно требовать применения эвристических правил. Задачи, которые могут быть гарантированно решены (с соблюдением заданных ограничений) с помощью некоторых формальных процедур, не подходят для применения ЭС;

3) задача должна быть достаточно сложна, чтобы оправдать затраты на разработку ЭС. Однако она не должна быть чрезмерно сложной (решение занимает у эксперта часы, а не недели), чтобы ЭС могла ее решать;

4) задача должна быть достаточно узкой, чтобы решаться методами ЭС, и практически значимой.

При разработке ЭС, как правило, используется концепция "быстрого прототипа". Суть этой концепции состоит в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип (прототипы) ЭС. Прототипы должны удовлетворять двум противоречивым требованиям: с одной стороны, они должны решать типичные задачи конкретного приложения, а с другой - время и трудоемкость их разработки должны быть весьма незначительны, чтобы можно было максимально запараллелить процесс накопления и отладки знаний (осуществляемый экспертом) с процессом выбора (разработки) программных средств (осуществляемым инженером по знаниям и программистом). Для удовлетворения указанным требованиям, как правило, при создании прототипа используются разнообразные средства, ускоряющие процесс проектирования.

Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера по знаниям расширяет знания прототипа о проблемной области. При неудаче может потребоваться разработка нового прототипа или разработчики могут прийти к выводу о непригодности методов ЭС для данного приложения. По мере увеличения знаний прототип может достигнуть такого состояния, когда он успешно решает все задачи данного приложения. Преобразование прототипа ЭС в конечный продукт обычно приводит к перепрограммированию ЭС на языках низкого уровня, обеспечивающих как увеличение быстродействия ЭС, так и уменьшение требуемой памяти. Трудоемкость и время создания ЭС в значительной степени зависят от типа используемого инструментария.

В ходе работ по созданию ЭС сложилась определенная технология их разработки, включающая шесть следующих этапов (рис. 1.4):

идентификацию, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию. На

этапе *идентификации* определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей.

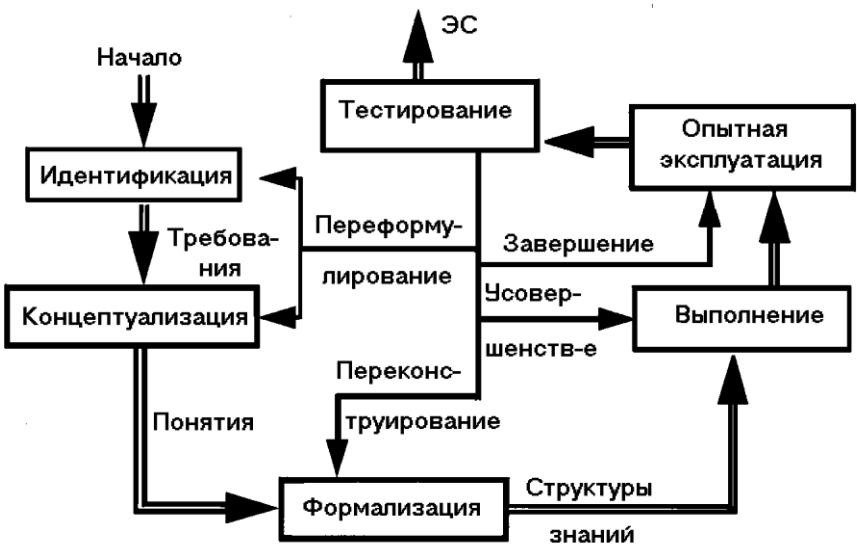



Рис.1.4. Технология разраб-ки ЭС.

На этапе *концептуализации* проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

На этапе *формализации* выбираются ИС и определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность целям системы зафиксированных понятий, методов решений, средств представления и манипулирования знаниями.

На этапе *выполнения* осуществляется наполнение экспертом базы знаний. В связи с тем, что основой ЭС являются знания, данный этап является наиболее важным и наиболее трудоемким этапом разработки ЭС. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, организацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа деятельности эксперта по решению реальных задач.

Интерфейс с конечным пользователем

Система G2 предоставляет разработчику богатые возможности для формирования простого, ясного и выразительного графического интерфейса с пользователем с элементами мультипликации. Предлагаемый инструментарий позволяет наглядно отображать технологические процессы практически неограниченной сложности на разных уровнях абстракции и детализации. Кроме того, графическое отображение взаимосвязей между объектами приложения может напрямую использоваться в декларативных конструкциях языка описания знаний.

RTworks не обладает собственными средствами для отображения текущего состояния управляемого процесса. Разработчик приложения вынужден использовать систему Dataview фирмы VI Corporation, что в значительной степени ограничивает его возможности.

Интерфейс с пользователем TDC Expert ограничен возможностями системы TDC 3000, т.е. взаимодействие с конечным пользователем

ограничивается текстовым режимом работы.

Представление знаний в экспертных системах

Первый и основной вопрос, который надо решить при представлении знаний, - это вопрос определения состава знаний, т.е. определение того, "ЧТО ПРЕДСТАВЛЯТЬ" в экспертной системе. Второй вопрос касается того, "КАК ПРЕДСТАВЛЯТЬ" знания. Необходимо отметить, что эти две проблемы не являются независимыми. Действительно, выбранный способ представления может оказаться непригодным в принципе либо неэффективным для выражения некоторых знаний.

По нашему мнению, вопрос "КАК ПРЕДСТАВЛЯТЬ" можно разделить на две в значительной степени независимые задачи: как организовать (структурировать) знания и как представить знания в выбранном формализме.

Стремление выделить организацию знаний в самостоятельную задачу вызвано, в частности, тем, что эта задача возникает для любого языка

представления и способы решения этой задачи являются одинаковыми (либо сходными) вне зависимости от используемого формализма.

Итак, в круг вопросов, решаемых при представлении знаний, будем включать следующие:

определение состава представляемых знаний;

организацию знаний;

представление знаний, т.е. определение модели представления. Состав знаний ЭС определяется следующими факторами:

проблемной средой;

архитектурой экспертной системы;

потребностями и целями пользователей;

языком общения.

В соответствии с общей схемой статической экспертной системы (см. рис. 1.1) для ее функционирования требуются следующие знания:

знания о процессе решения задачи (т.е. управляющие знания), используемые интерпретатором (решателем);

знания о языке общения и способах организации диалога, используемые лингвистическим процессором (диалоговым компонентом);

знания о способах представления и модификации знаний, используемые компонентом приобретения знаний;

поддерживающие структурные и управляющие знания, используемые объяснительным компонентом.

Для динамической ЭС, кроме того, необходимы следующие знания:

1) знания о методах взаимодействия с внешним окружением;

2) знания о модели внешнего мира.

Зависимость состава знаний от требований пользователя проявляется в следующем:

какие задачи (из общего набора задач) и с какими данными хочет решать пользователь;

каковы предпочтительные способы и методы решения;

при каких ограничениях на количество результатов и способы их получения должна быть решена задача;

каковы требования к языку общения и организации диалога;

какова степень общности (конкретности) знаний о проблемной области, доступная пользователю;

каковы цели пользователей.

Состав знаний о языке общения зависит как от языка общения, так и от требуемого уровня понимания.

С учетом архитектуры экспертной системы знания целесообразно делить на *интерпретируемые* и *неинтерпретируемые* . К первому типу относятся те знания, которые способен интерпретировать решатель (интерпретатор). Все остальные знания относятся ко второму типу. Решатель не знает их структуры и содержания. Если эти знания используются каким-либо компонентом системы, то он не "осознает" этих знаний. Неинтерпретируемые знания подразделяются на*вспомогательные* знания, хранящие информацию о лексике и грамматике языка общения, информацию о структуре диалога, и *поддерживающие* знания. Вспомогательные знания обрабатываются естественно-языковой компонентой, но ход этой обработки решатель не осознает, так как этот этап обработки входных сообщений является вспомогательным для проведения экспертизы. Поддерживающие знания используются при создании системы и при выполнении объяснений. Поддерживающие знания выполняют роль описаний (обоснований) как интерпретируемых знаний, так и действий системы. Поддерживающие знания подразделяются на *технологические* и *семантические***.** Технологические поддерживающие знания содержат сведения о времени создания описываемых ими знаний, об авторе знаний и т.п. Семантические поддерживающие знания содержат смысловое описание этих знаний. Они содержат информацию о причинах ввода знаний, о назначении знаний, описывают способ использования знаний

и получаемый эффект. Поддерживающие знания имеют описательный характер.

Интерпретируемые знания можно разделить на *предметные знания, управляющие знания* и *знания о представлении.* Знания о представлении содержат информацию о том, каким образом (в каких структурах) в системе представлены интерпретируемые знания.

Предметные знания содержат данные о предметной области и способах преобразования этих данных при решении поставленных задач. Отметим, что по отношению к предметным знаниям знания о представлении и знания об управлении являются *метазнаниями.* В предметных знаниях можно выделить описатели и собственно предметные знания. Описатели содержат определенную информацию о предметных знаниях, такую, как коэффициент определенности правил и данных, меры важности и сложности. Собственно предметные знания разбиваются на *факты* и *исполняемые утверждения.* Факты определяют возможные значения сущностей и характеристик предметной области. Исполняемые утверждения содержат информацию о том, как можно изменять описание предметной области в ходе решения задач. Говоря другими словами, исполняемые *утверждения* - это знания, задающие процедуры обработки. Однако мы избегаем использовать термин "процедурные знания", так как хотим подчеркнуть, что эти знания могут быть заданы не только в процедурной, но и в декларативной форме.

Управляющие знания можно разделить на *фокусирующие* и *решающие.* Фокусирующие знания описывают, какие знания следует использовать в той или иной ситуации. Обычно фокусирующие знания содержат сведения о наиболее перспективных объектах или правилах, которые целесообразно использовать при проверке соответствующих гипотез (см. п. 9.2). В первом случае внимание фокусируется на элементах рабочей памяти, во втором - на правилах базы знаний. Решающие знания содержат информацию, используемую для выбора

способа интерпретации знаний, подходящего к текущей ситуации. Эти знания применяются для выбора стратегий или эвристик, наиболее эффективных для решения данной задачи.

Качественные и количественные показатели экспертной системы могут быть значительно улучшены за счет использования *метазнании,* т.е. знаний о знаниях. Метазнания не представляют некоторую единую сущность, они могут применяться для достижения различных целей. Перечислим возможные назначения метазнаний :

1) метазнания в виде стратегических метаправил используются для выбора релевантных правил ;

2) метазнания используются для обоснования целесообразности применения правил из области экспертизы;

3) метаправила используются для обнаружения синтаксических и семантических ошибок в предметных правилах;

4) метаправила позволяют системе адаптироваться к окружению путем перестройки предметных правил и функций;

5) метаправила позволяют явно указать возможности и ограничения системы, т.е. определить, что система знает, а что не знает.

Вопросы организации знаний необходимо рассматривать в любом представлении, и их решение в значительной степени не зависит от выбранного способа (модели) представления. Выделим следующие аспекты проблемы организации знаний :

организация знаний по уровням представления и по уровням детальности;

организация знаний в рабочей памяти;

организация знаний в базе знаний.

## 2. **Общие сведения о графах**

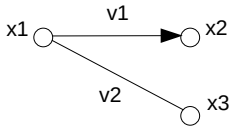
*В последнее время наблюдается неуклонное вторжение математических методов в различные отрасли науки и техники.*

*Теория графов изучает графы как абстрактные математические образования, независимо от их конкретных истолкований, а полученные общие результаты затем прилагаются к самым различным дисциплинам.*

*Как математическая дисциплина теория графов сформировалась в середине XX в., хотя отдельные задачи о графах имеют 200-летнюю давность. Термин «граф» приобрел право гражданства и вошел в математический язык в 1936 г., после выхода в свет монографии Кёнига, в которой впервые графы изучаются как самостоятельные математические объекты независимо от их содержания. За последние 25 лет теория графов развивается весьма быстро. В связи с интенсивным развитием экономико-математических исследований возникает много задач и методов связанных с графами.*

*Граф* – совокупность объектов (X, V), где X – множество вершин графа, а V – множество связей графа, символически обозначаемый G (X,У).

Если связь между вершинами не направлена, то ее называют *ребром***.** Если связь между вершинами направлена, то ее называют *дугой.*




где связь v1 является дугой, а v2 - ребром.

Если в графе имеется ребро (дуга) ($x_i$, $x_j$) (причем для дуги $x_i$ - вершина, из которой она исходит, $x_j$ - в которую заходит), то говорят, что вершины $x_i$ и $x_j$ *смежные***,** и *не смежные* в противном случае.

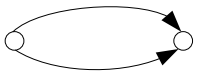
Вершины, определяющие связь, называются *концевыми* вершинами этой связи.

Два ребра (дуги) называются смежными**,** если они имеют общую концевую вершину.

Ребро (дуга) и ее концевые вершины являются *инцидентными* друг другу.

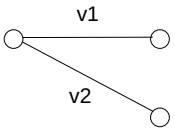
Смежность есть отношение между однородными элементами графа, тогда как инцидентность является отношением между разнородными элементами.

Все ребра (дуги) с одинаковыми концевыми вершинами (дуги должны иметь одинаковое направление) называются *параллельными или кратными*.



Ребро (дуга) с совпадающими концевыми вершинами называется *петлей.*

Граф называется *простым*, если он не содержит петель и кратных *ребер* (дуг).




Простой граф называется *полным*, если в нем каждая пара вершин является смежной.

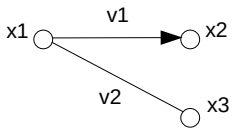
Граф, содержащий кратные связи, называет *мультаграфом*.

Если в графе допускаются и петли, и кратные связи, то граф называется *псевдографом*.

Граф, не имеющий связей, называется *пустым*.

Если в графе все связи направлены (дуги), то такой граф называется *ориентированным* (орграфом), в противном случае *неориентированным*.

Если в графе присутствуют ребра и дуги, то такой граф является *смешанным*.



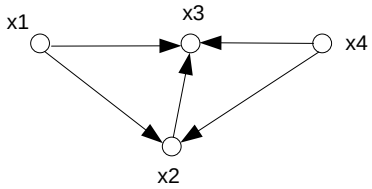

*Степенью вершины* $P(x_i)$ называется число ребер графа G инцидентных вершине $x_i$.

Вершина графа, имеющая степень 0 называется *изолированной*, а степень 1 *висячей*.

Для орграфа вводятся характеристики *полустепень исхода* $P^-(x_i)$ - количество дуг исходящих из вершины $x_i$ и *полустепень захода* $P^+(x_i)$ - количество дуг заходящих в вершину $x_i$.

Для ориентированного графа справедливо равенство $P(x_i) = P^+(x_i) + P^-(x_i)$.

Пр.



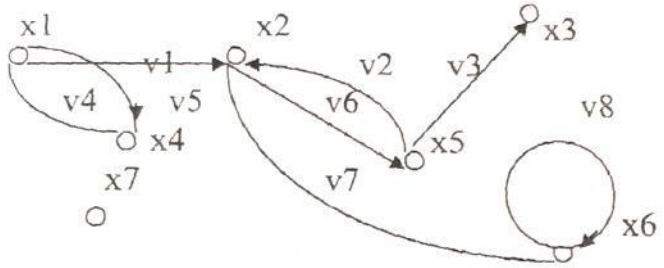

$P(x_i) = P^+(x_i) + P^-(x_i)$
$P(x_1) = 0 + 2 = 2$
$P(x_2) = 2 + 1 = 3$
$P(x_3) = 3 + 0 = 3$
$P(x_4) = 0 + 2 = 2$

## Способы задания графов

Существуют четыре способа задания графов: графический, перечислительный, с помощью матриц смежности и инцидентности.

1. *Графический способ*

Граф задается множеством вершин и связей:




(1)

*2. Перечислительный способ*

Граф задается перечислением множества вершин, множества дуг и ребер, списком изолированных вершин.

Граф, изображенный на рис., можно задать следующим описанием:

множество вершин: $X = \{x_1, x_2, x_3, x_3, x_5, x_6, x_7\}$;

множество дуг: $V = \{(x_1, x_2),(x_5, x_2),(x_5, x_3),(x_1, x_4),(x_2, x_5),(x_6, x_6)\}$;

множество ребер: $V = \{(x_1, x_4),(x_2, x_6)\}$;

список изолированных вершин: {X7}.

3. *Матрица смежности*

Матрица смежности является квадратной матрицей размером n x n, где n – количество вершин заданного графа $A(G) = \|a_{ij}\|$.

Для неориентированного графа $a_{ij}$, принимает следующие значения:

$$a_{ij} = \begin{cases} k, \text{где } k - кол - во \text{ связей между } x_i \text{ и } x_j \text{ вершинами} \\ 0, \text{если связи между } x_i \text{ и } x_j \text{ отсутствуют} \end{cases}$$

Для ориентированного графа $a_{ij}$, принимает следующие значения:

$$a_{ij} = \begin{cases} k, \text{где } k - кол - во \text{ дуг исходящих из } x_i \text{ и заходящих в } x_j \\ 0, \text{если связи между } x_i \text{ и } x_j \text{ отсутствуют} \end{cases}$$

По матрице смежности можно определить:

1. полустепень исхода вершины $x_i$ как сумму чисел в i-строке;
2. полустепень захода вершины $x_i$ как сумму чисел в i-столбце;
3. петли как элемнты главной диагонали матрицы;

4. изолированные вершины при условии, что соответствующие строки и столбцы содержат только нулевые элементы.

Пример:

Построим матрицу смежности для графа (1):

| $x_i / x_j$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 1 | 0 | 2 | 0 | 0 | 0 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| $x_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $x_4$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $x_5$ | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| $x_6$ | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| $x_7$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

4. *Матрица инцидентности*

Матрица инцидентности – это матрица размером nxm, где n – количество вершин графа, а m - количество связей графа $B(G) = \|b_{ij}\|$.

Для неориентированного графа $b_{ij}$, принимает следующие значения:

$$b_{ij} = \begin{cases} 1, \text{если вершина } x_i \text{ инцидентна ребру } v_j \\ 0, \text{если вершина } x_i \text{ не инцидентна ребру } v_j \end{cases}$$

Для ориентированного графа $b_{ij}$, принимает следующие значения:

$$b_{ij} = \begin{cases} 1, \text{если вершина } x_i \text{ начало дуги } v_j \\ -1, \text{если вершина } x_i \text{ конец дуги } v_j \\ 0, \text{если вершина } x_i \text{ не инцидентна дуге } v_j \end{cases}$$

По матрице инцидентности можно определить:

1. если в столбце 1-а единица, то столбец петля;
2. если единицы имеют противоположные знаки, то граф ориентирован.

Пример:

Построим матрицу инцидентности для графа (1):

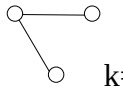
| $x_i / v_j$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| $x_2$ | -1 | -1 | 0 | 0 | 0 | 1 | 1 | 0 |
| $x_3$ | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| $x_4$ | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 |
| $x_5$ | 0 | 1 | 1 | 0 | 0 | -1 | 0 | 0 |
| $x_6$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| $x_7$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## Связность. Компоненты связности

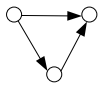
Вершина $x_1$ графа G достижима из вершины $x_2$, если $x_1 = x_2$, либо существует маршрут соединяющий вершины $x_1$, $x_2$.

Граф (орграф) называется связным (сильно связным), если для любых его вершин $x_i$, $x_j$ существует маршрут, соединяющий $x_i$ и $x_j$.



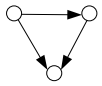
k=1

Граф (орграф) называется односторонне связным, если для любых двух его вершин хотя бы одна вершина достижима из другой.



k=1

Орграф называется слобосвязным, если полученный из него не ориентированный граф, путем замены дуг на ребра является связным.



k=1

Максимальный по включению вершин связный (сильно связный) подграф графа называется его компонентой связности (k).

Общее число компонент связности графа называется степенью связности графа.

Матрица связности графа G называется квадратная матрица $S(G) = \|s_{ij}\|$

$$s_{ij} = \begin{cases} 1, \text{если i = j или существует маршрут соединяющий две вершины} \\ 0, \text{иначе} \end{cases}$$

Матрица сильной связности графа G называется квадратная матрица

$$S(G) = \|s_{ij}\|$$

$$s_{ij} = \begin{cases} 1, \text{если одна вершина достижима из другой и обратно} \\ 0, \text{иначе} \end{cases}$$

Алгоритм определения компоненты связности:

1.Составим две матрицы:

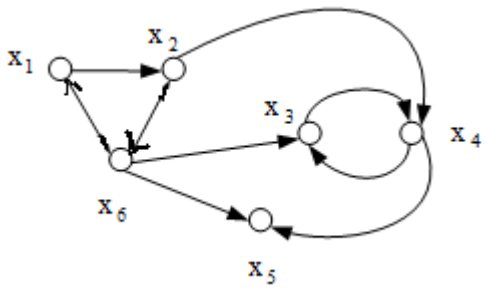
- Матрицу смежности;
- Матрицу связности.

2. В матрице $S(G)$ в 1-ой строке находим единицы и заносим в 1-ую компоненту связности.

Для того, что бы определить как будут связаны полученные вершины нужно из матрицы $A(G)$ выбрать те строки и столбцы, которые соответствуют взятым вершинам.

3. Вычеркиваем из обеих матриц $S(G)$ и $A(G)$ строки и столбцы, которые соответствуют найденным вершинам в п.2.
4. Повторяем п.2 и п.3 до тех пор пока не вычеркнутая все строки и столбцы.

Пример:




1. Построим матрицу смежности для графа:

| $A(G)$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
|---|---|---|---|---|---|---|
| $x_1$ | 0 | 1 | 0 | 0 | 0 | 0 |
| $x_2$ | 0 | 0 | 0 | 1 | 0 | 1 |
| $x_3$ | 0 | 0 | 0 | 1 | 0 | 0 |

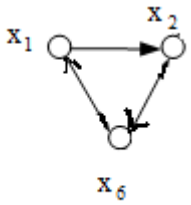
| | | | | | | |
|---|---|---|---|---|---|---|
| $x_4$ | 0 | 0 | 1 | 0 | 1 | 0 |
| $x_5$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $x_6$ | 1 | 0 | 1 | 0 | 1 | 0 |

2. Построим матрицу связности для графа:

| $S(G)$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
|---|---|---|---|---|---|---|
| $x_1$ | 1 | 1 | 0 | 0 | 0 | 1 |
| $x_2$ | 1 | 1 | 0 | 0 | 0 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 |
| $x_4$ | 0 | 0 | 1 | 1 | 0 | 0 |
| $x_5$ | 0 | 0 | 0 | 0 | 1 | 0 |
| $x_6$ | 1 | 1 | 0 | 0 | 0 | 1 |

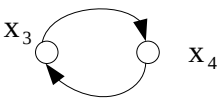
3. В матрице $S(G)$ в 1-ой строке находим единицы и заносим в 1-ую компоненту связности: $k_1=\{ x_1, x_2, x_6 \}$

Для того, что бы определить как будут связаны полученные вершины нужно из матрицы $A(G)$ выбрать те строки и столбцы, которые соответствуют взятым вершинам.




| $A(G)$ | $x_1$ | $x_2$ | $x_6$ |
|---|---|---|---|
| $x_1$ | 0 | 1 | 0 |
| $x_2$ | 0 | 0 | 1 |
| $x_6$ | 1 | 0 | 0 |

4. В матрице $S(G)$ в 3-ей строке находим единицы и заносим во 2-ую компоненту связности: $k_2=\{ x_3, x_4 \}$.




| $A(G)$ | $x_3$ | $x_4$ |
|---|---|---|
| $x_3$ | 0 | 1 |
| $x_4$ | 1 | 0 |

5. В матрице $S(G)$ в 5-ой строке находим единицы и заносим в 3-ую компоненту связности: $k_3=\{ x_5 \}$.

Ответ: K=3

Цикломатическое число вычисляется по формуле: $\nu(G) = m - n + k$, где m – число связей; n - число вершин; k – коэффициент связности.

## Метрические характеристики графа

Длина кратчайшего пути маршрута называется расстоянием между вершинами $d(x_i, x_j)$.

Диаметром графа G называется $\max (d(x_i, x_j))$.

Пусть $x_i$ - произвольная вершина из X. Величина $r(x)=\max d(x_i, x_j)$ называется максимальным удалением (эксцентриситет).

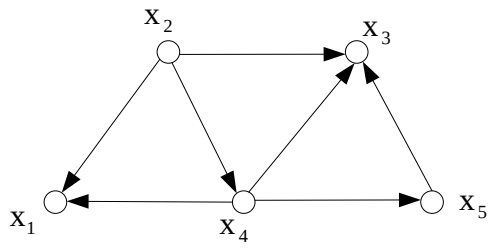
Радиусом графа G называется величина $r(G)=\min (r(x_i))$.

Любая вершина $x_i \in X$ такая, что $r(x_i)= r(G)$ называется центром графа G (центральные вершины).

Задача нахождения центральных вершин графа постоянно используется в практической деятельности людей.

Пусть, например, граф представляет сеть дорог, т.е. вершины его соответствуют отдельным населенным пунктам, а ребра – дорогам между ними. Требуется оптимально разместить магазины, пункты обслуживания, АЗС. В подобных ситуациях критерий оптимальности часто заключается в оптимальности «наихудшего» случая, т.е. в минимальном расстоянии от места обслуживания до наиболее удаленного пункта. Следовательно, местами размещения должны быть центральные вершины.

Пример:




Решение:

1. Расстояние между вершинами:

$d(x_1, x_2)=1$

$d(x_1, x_3)=2$ $d(x_1, x_3)=2$

$d(x_1, x_4)=1$ $d(x_2, x_4)=1$ $d(x_3, x_4)=1$

$d(x_1, x_5)=2$ $d(x_2, x_5)=2$ $d(x_3, x_5)=1$ $d(x_4, x_5)=1$

2. Диаметром графа: $d(G)=\max (d(x_i, x_j))=2$
3. Максимальным удалением (эксцентриситет):

   $r(x_1)=2$

   $r(x_2)=2$

   $r(x_3)=2$

   $r(x_4)=1$

   $r(x_5)=2$
4. Радиусом графа: $r(G)=\min (r(x_i))=1$
5. Центральная вершина: $x_4$

## Алгоритм Дейкстры

Предположим, что необходимо посетить каждый из нескольких городов (магазинов, складов и т.д.) и возвратиться к своему первоначальному положению. Учитывая сеть дорог, соединяющих различные города (магазины и т.д.) на маршруте, проблема состоит в том, чтобы найти маршрут, который минимизирует полное расстояние путешествия.

Сеть дорог может быть представлена взвешенным графом (т.е. графом каждому ребру, которого присвоено определенное число – вес ребра) следующим образом. Каждый интересующий нас объект представляется вершиной, а каждая дорога или маршрут – ребром, соединяющим соответствующие две вершины, причем вес ребра равен длине данной дороги. Путешествие, при котором посещается каждый объект и которое заканчивается в исходном (стартовом) положении, представляется замкнутой последовательностью ребер графа, у которого последовательность вершин содержит все вершины. В терминологии теории графов цикл графов, проходящий через все вершины графа ровно по одному разу называется гамильтоновым, при этом необязательно проходить через все ребра графа. Вес гамильтонова цикла – сумма весов ребер, содержащихся в нем.

Как правило, оказывается целесообразным несколько видоизменить исходную проблему следующим образом. Каждому ребру полного графа (*граф, в котором все вершины попарно соединены между собой*) приписывается вес, равный наикратчайшему расстоянию между соответствующими объектами, в соответствии с используемой сетью дорог. Эти самые короткие расстояния могут быть найдены с помощью алгоритмов Декстра, Форда-Беллмана.

## Описание алгоритма Дейкстры

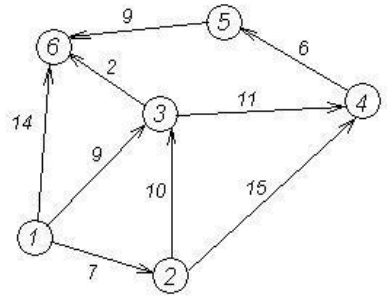
Метка самой вершины *a* полагается равной *0*, метки остальных вершин — бесконечности. Это отражает то, что расстояния от *a* до других вершин пока неизвестны.

Все вершины графа помечаются как «непосещенные».

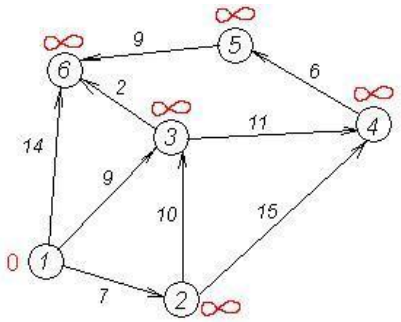
Если все вершины посещены, алгоритм завершается. В противном случае из еще не посещенных вершин выбирается вершина *u*, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых *u* является предпоследним пунктом. Вершины, соединенные с вершиной *u* ребрами, назовем соседями этой вершины. Для каждого соседа рассмотрим новую длину пути, равную сумме текущей метки *u* и длины ребра, соединяющего *u* с этим соседом. Если полученная длина меньше метки соседа, заменим метку этой длиной. Рассмотрев всех соседей, пометим вершину *u* как посещенную и повторим шаг.

Пример

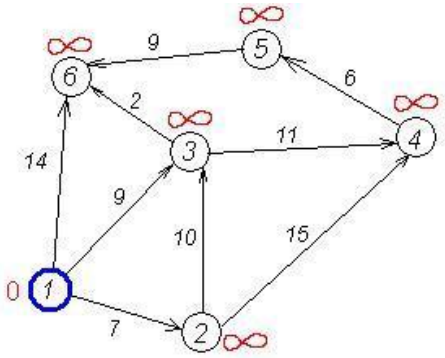
Рассмотрим работу алгоритма на примере графа, показанного на рисунке. Пусть требуется найти расстояния от 1-й вершины до всех остальных.



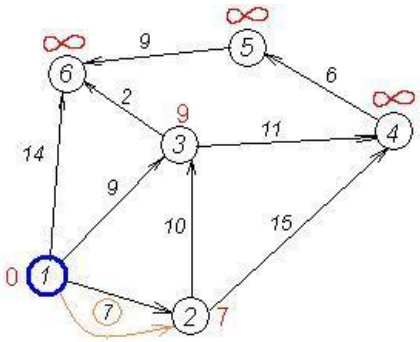

Кружками обозначены вершины, линиями — пути между ними (ребра графа). В кружках обозначены номера вершин, над ребрами обозначена их «цена» — длина пути. Рядом с каждой вершиной красным обозначена метка — длина кратчайшего пути в эту вершину из вершины 1.



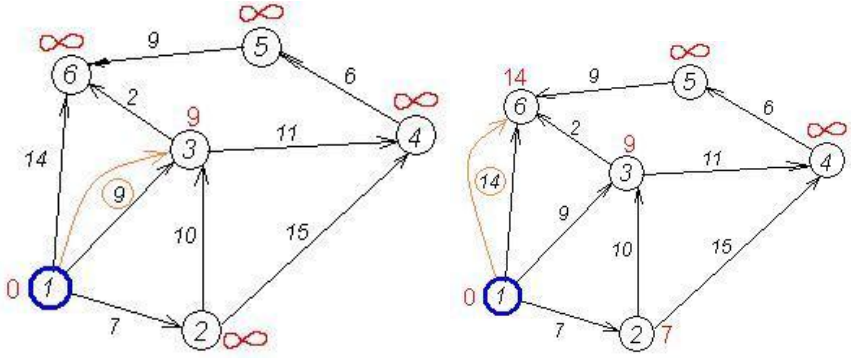
Первый шаг: Рассмотрим шаг алгоритма Дейкстры для нашего примера. Минимальную метку имеет вершина 1. Ее соседями являются вершины 2, 3 и 6.



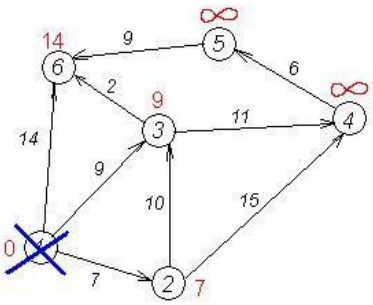

Первый по очереди сосед «вершины 1» — «вершина 2», потому что длина пути до нее минимальна. Длина пути в нее через «вершину 1» равна кратчайшему расстоянию до «вершины 1» + длина ребра, идущего из 1 в 2, то есть 0 + 7 = 7. Это меньше текущей метки вершины 2, поэтому новая метка 2-й вершины равна 7.



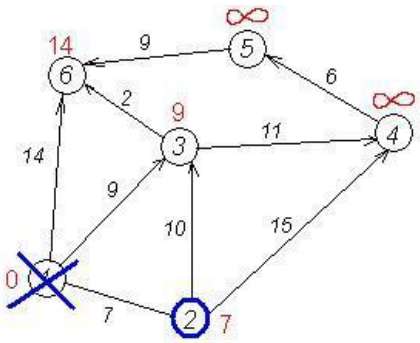

Аналогичную операцию проделываем с двумя другими соседями 1-й вершины — 3-й и 6-й.




Все соседи вершины 1 проверены. Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру не подлежит. Вычеркнем её из графа, чтобы отметить, что эта вершина посещена.




Второй шаг: Шаг алгоритма повторяется. Снова находим минимальную метку из непосещенных вершин. Это вершина 2 с меткой 7.

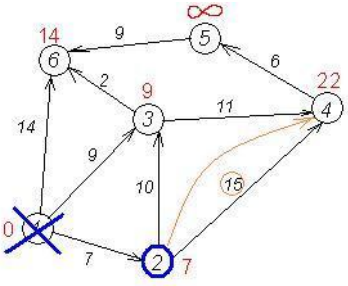



Снова пытаемся уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2-ю. Соседями вершины 2 являются 1, 3, 4.

Первый (по порядку) сосед вершины 2 — вершина 1. Но она уже посещена, поэтому с 1-й вершиной ничего не делаем.

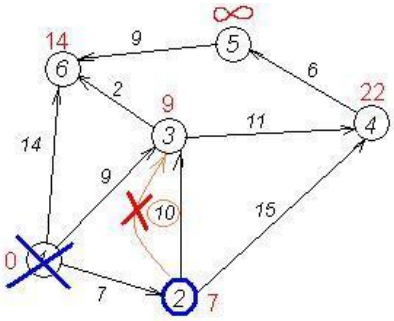
Следующий сосед вершины 2 — вершина 4.

Длина пути будет = кратчайшее расстояние до 2 + расстояние между вершинами 2 и 4 = 7 + 15 = 22. Т.к. 22<∞, устанавливаем метку вершины 4 равной 22.




Ещё один сосед вершины 2 — вершина 3.

Длина такого пути будет = 7 + 10 = 17. Но текущая метка третьей вершины равна 9<17, поэтому метка не меняется.



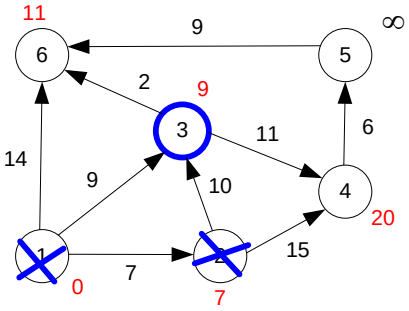

Все соседи вершины 2 просмотрены, замораживаем расстояние до неё и помечаем ее как посещенную.

Третий шаг: Снова находим минимальную метку из непосещенных вершин (вершина 3).

Соседями вершины 3 являются 6 и 4.

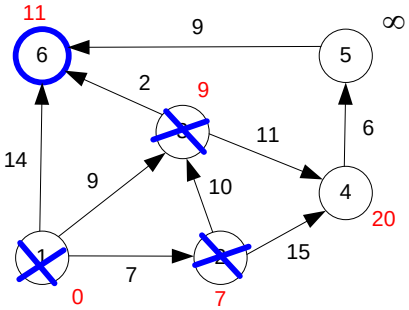
вершина 4: длина пути будет = 9 + 11 = 20. Текущая метка вершины 4 равна 22, т.к. 20<22 меняем метку на 20.

вершина 6: длина пути будет = 9 + 2 = 11. Текущая метка вершины 6 равна 14, т.к. 11<14 меняем метку на 11.



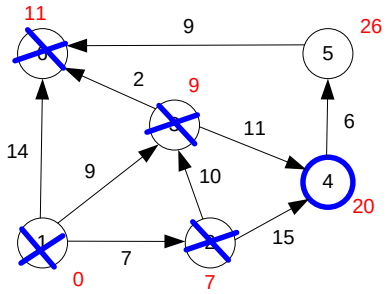

Все соседи вершины 3 просмотрены, замораживаем расстояние до неё и помечаем ее как посещенную.

Четвертый шаг: Снова находим минимальную метку из непосещенных вершин (вершина 6). У данной вершины соседей нет, соответственно помечаем ее как посещенную.



Пятый шаг: Снова находим минимальную метку из непосещенных вершин (вершина 4). Соседом вершины 4 являются 5.

вершина 5: длина пути будет = 20 + 6 = 26. Т.к. 26<∞, устанавливаем метку вершины 5 равной 26.




Все соседи вершины 4 просмотрены, замораживаем расстояние до неё и помечаем ее как посещенную.

Завершение выполнения алгоритма: Алгоритм заканчивает работу, когда вычеркнуты все вершины. Результат его работы: кратчайший путь от вершины 1 до 2-й составляет 7, до 3-й — 9, до 4-й — 20, до 5-й — 26, до 6-й — 11.

## 3. Транспортные задачи

Транспортная задача является одной из наиболее распространенных специальных задач экспертного линейного программирования. Частные постановки задачи рассмотрены рядом специалистов по транспорту, например О. Н. Толстым. Первая строгая постановка транспортной задачи принадлежит Ф. Хичкоку, поэтому в зарубежной литературе ее называют проблемой Хичкока. Первый точный метод решения Т-задачи разработан Л. В. Канторовичем и М. К. Гавуриным. Под названием “транспортная задача” объединяется широкий круг задач с единой математической моделью.

**Постановка задачи**

Под термином "транспортные задачи" понимается широкий круг задач не только транспортного характера. Общим для них является, как правило, распределение ресурсов, находящихся у *m* производителей (поставщиков), по *n* потребителям этих ресурсов. В качестве критериев в транспортных задачах используют: *критерий стоимости* (план перевозок оптимален, если достигнут минимум затрат на его реализацию), *критерий времени* (план оптимален, если на его реализацию затрачивается минимум времени) и другие.

Наиболее часто встречаются следующие задачи, относящиеся к транспортным:

- прикрепление потребителей ресурса к производителям;
- привязка пунктов отправления к пунктам назначения;
- взаимная привязка грузопотоков прямого и обратного направлений;
- отдельные задачи оптимальной загрузки промышленного оборудования;
- оптимальное распределение объемов выпуска промышленной продукции между заводами-изготовителями и др.

Рассмотрим экономико-математическую модель прикрепления пунктов отправления к пунктам назначения. Имеются $m$ пунктов отправления груза и объемы отправления по каждому пункту $a_1$, $a_2$ ,...,$a_m$ . Известна потребность в грузах $b_1$, $b_2$ ,...,$b_n$ по каждому из $n$ пунктов назначения. Задана матрица стоимостей доставки по каждому варианту $c_{ij}$, $i=\overline{1,m}$ $j=\overline{1,n}$. Необходимо рассчитать оптимальный план перевозок, т.е. определить, сколько груза должно быть отправлено из каждого $i$-го пункта отправления (от поставщика) в каждый $j$-ый пункт назначения (до потребителя) $x_{ij}$ с минимальными транспортными издержками.

В общем виде исходные данные представлены в табл. 3.1. Строки транспортной таблицы соответствуют пунктам отправления (в последней клетке каждой строки указан объем запаса продукта $a_i$ ), а столбцы - пунктам назначения (последняя клетка каждого столбца содержит значение потребности $b_j$). Все клетки таблицы (кроме тех, которые расположены в нижней строке и правом столбце) содержат информацию о перевозке из $i$-го пункта в $j$-й: в правом верхнем углу находится цена перевозки единицы продукта, а в левом нижнем - значение объема перевозимого груза для данных пунктов.

Таблица 1. Исходные данные

| Потребители / Поставщики | $B_1$ | $B_2$ | … | $B_n$ | Запасы (объемы отправления) |
|---|---|---|---|---|---|
| $A_1$ | $c_{11}$ $x_{11}$ | $c_{12}$ $x_{12}$ | … | $c_{1n}$ $x_{1n}$ | $a_1$ |
| $A_2$ | $c_{21}$ $x_{21}$ | $c_{22}$ $x_{22}$ | … | $c_{2n}$ $x_{2n}$ | $a_2$ |
| … | … | … | … | … | … |
| $A_m$ | $c_{m1}$ $x_{m1}$ | $c_{m2}$ $x_{m2}$ | … | $c_{mn}$ $x_{mn}$ | $a_m$ |
| Потребность | $b_1$ | $b_2$ | … | $b_n$ | |

Транспортная задача называется закрытой, если суммарный объем отправляемых грузов $\sum_{i=1}^{m} a_i$ .равен суммарному объему потребности в этих грузах по пунктам назначения $\sum_{j=1}^{n} b_i$

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_i \qquad 3.1)$$

Если такого равенства нет (потребности выше запасов или наоборот), запасу называют открытой, т.е.:

$$\sum_{i=1}^{m} a_i \neq \sum_{j=1}^{n} b_i \qquad (3.2)$$

Транспортным задачам присущи следующие особенности:

- распределению подлежат однородные ресурсы;
- условия задачи описываются только уравнениями;
- все переменные выражаются в одинаковых единицах измерения;
- во всех уравнениях коэффициенты при неизвестных равны единице;
- каждая неизвестная встречается только в двух уравнениях системы ограничений.

Опорный план является допустимым решением транспортной задачи и используется в качестве начального базисного решения при нахождении оптимального решения методом потенциалов. Существует три метода нахождения опорных планов: метод северо-западного угла, метод минимального элемента и метод Фогеля. "Качество" опорных планов, полученных этими методами, различается: в общем случае метод Фогеля дает наилучшее решение (зачастую оптимальное), а метод северо-западного угла – наихудшее.

Все существующие методы нахождения опорных планов отличаются только способом выбора клетки для заполнения. Само заполнение происходит одинаково независимо от используемого метода.

**Методы составления начального опорного плана**

Базисный план составляется последовательно, в несколько шагов (точнее, $m + n - 1$ шагов). На каждом из этих шагов заполняется одна клетка, притом так, что, полностью удовлетворяется один из заказчиков (тот, в столбце которого находится заполняемая клетка), либо полностью вывозится весь запас груза с одной из баз (с той, в строке которой находится заполняемая клетка).

Начиная с первоначально данной таблицы и повторив ($m + n - 2$) раз описанный шаг, мы придем к “таблице”, состоящей из одной строки и одного столбца (иначе говоря, из одной пустой клетки). Другими словами, мы пришли к задаче с одной базой и с одним потребителем, причем потребности этого единственного заказчика равны запасу груза на этой единственной базе. Заполнив последнюю клетку, мы освобождаем последнюю базу и удовлетворяем потребность последнего заказчика. В результате, совершив ($m + n - 1$) шагов, мы и получим искомый опорный план.

**Замечание:** Может случиться, что уже на некотором (но не на последнем!) шаге потребность очередного заказчика окажется равной запасу груза на очередной базе. Тогда после заполнения очередной клетки объем таблицы как бы одновременно уменьшается на одни столбец и на одну строку. Но и при этом мы должны считать, что уменьшение объема таблицы происходит либо на один столбец, а на базе сохраняется "остаток" равный нулю, либо на одну строку, а у заказчика еще осталась неудовлетворенная "потребность" в количестве нуля единиц груза, которая и удовлетворяется на одном из следующих шагов. Этот нуль ("запас" или "потребностью" – безразлично) надо записать в очередную заполняемую клетку на одном из последующих шагов. Так как при этом оказывается равной нулю одна из базисных неизвестных, то мы имеем дело с вырожденным случаем.

**1. Метод северо-западного угла.**

При этом методе на каждом шаге построения первого опорного плана заполняется левая верхняя клетка (северо-западный угол) оставшейся части таблицы. При таком методе заполнение таблицы начинается с клетки неизвестного $x_{11}$ и заканчивается в клетке неизвестного $x_{mn}$ , т. е. идет как бы по диагонали таблицы перевозок.

Пример:

| Пункты Отправления | Запасы | Пункты назначения | | | | |
|---|---|---|---|---|---|---|
| | | $B_1$ | $B_2$ | $B_3$ | $B_4$ | $B_5$ |
| | | 170 | 110 | 100 | 120 | 200 |
| $A_1$ | 300 | 70 **170** | 50 **110** | 15 **20** | 80 | 70 |
| $A_2$ | 150 | 80 | 90 | 40 **80** | 60 **70** | 85 |
| $A_3$ | 250 | 50 | 10 | 90 | 11 **50** | 25 **200** |

1. Заполнение таблицы начинается с ее северо-западного угла, т.е. клетки с неизвестным $x_{11}$.
2. Первая база $A_1$ может полностью удовлетворить потребность первого заказчика $B_1$ ($a_1$=300, $b_1$=170, $a_1 > b_1$). Полагая $x_{11}$= 170, вписываем это значение в клетку $x_{11}$ и исключаем из рассмотрения первый столбец. На базе $A_1$ остается измененный запас $a'_1 = 130$.
3. В оставшейся новой таблице с тремя строками $A_1$,$A_2$,$A_3$ и четырьмя столбцами $B_1$,$B_2$,$B_3$,$B_4$; северо-западным углом будет клетка для неизвестного $x_{12}$ . Первая база с запасом $a'_1 = 130$может полностью удовлетворить потребность второго заказчика $B_2$ $(a'_1 = 130, b_2 = 110, a'_1 > b_2)$. Полагаем $x_{12}$ = 110, вписываем это значение в клетку $x_{12}$ и исключаем из рассмотрения второй столбец. На базе $A_1$ остается новый остаток (запас) $a''_1 = 20$.

4. В оставшейся новой таблице с тремя строками $A_1$,$A_2$,$A_3$ и тремя столбцами $B_3$,$B_4$,$B_5$ северо-западным углом будет клетка для неизвестного $x_{13}$. Теперь третий заказчик $B_3$ может принять весь запас с базы $A_1$ $(a''_1 = 20, b_3 = 100, a''_1 < b_3)$. Полагаем $x_{13}$ = 20, вписываем это значение в клетку $x_{13}$ и исключаем из рассмотрения первую строку. У заказчика из $B_3$ осталась еще не удовлетворенной потребность $b'_3 = 80$.

5. Теперь переходим к заполнению клетки для неизвестного $x_{23}$ и т.д. Через шесть шагов у нас останется одна база $A_3$ с запасом груза (остатком от предыдущего шага) $a'_3 = 200$ и один пункт $B_5$ с потребностью $b_5$=200 . Соответственно этому имеется одна свободная клетка, которую и заполняем, положив $x_{35}$=200. План составлен.

6. Базис образован неизвестными $x_{11}$,$x_{12}$,$x_{13}$,$x_{23}$,$x_{24}$,$x_{34}$,$x_{35}$. Правильность составленного плана легко проверить, подсчитав суммы чисел, стоящих в заполненных клетках по строкам и столбцам.

7. Общий объем перевозок в тонно-километрах для этого плана составит:

$$S_1 = 70 \cdot 170 + 50 \cdot 110 + 15 \cdot 20 + 40 \cdot 80 + 60 \cdot 70 + 11 \cdot 50 + 25 \cdot 200 = 30650$$

**2. Метод наименьшей стоимости**.

При этом методе на каждом шаге построения опорного плана первою заполняется та клетка оставшейся части таблицы, которая имеет наименьший тариф. Если такая клетка не единственная, то заполняется любая из них.

***Пример***

| Пункты Отправления | Запасы | Пункты назначения | | | | |
|---|---|---|---|---|---|---|
| | | $B_1$ | $B_2$ | $B_3$ | $B_4$ | $B_5$ |
| | | 170 | 110 | 100 | 120 | 200 |
| $A_1$ | 300 | 70 **20** | 50 | 15 **100** | 80 | 70 **180** |
| $A_2$ | 150 | 80 **150** | 90 | 40 | 60 | 85 |
| $A_3$ | 250 | 50 | 10 **110** | 90 | 11 **120** | 25 **20** |

1. В данном случае заполнение таблицы начинается с клетки для неизвестного $x_{32}$, для которого мы имеем значение $c_{32}$ = 10, наименьше из всех значений $c_{ij}$ . Эта клетка находится на пересечении третьей строки и второго столбца, соответствующим третьей базе $A_3$ и второму заказчику $B_2$.
2. Третья база $A_3$ может полностью удовлетворить потребность второго заказчика $B_2$ ($a_3$=250, $b_2$=110, $a_3 > b_2$) . Полагая $x_{32}$ = 110, вписываем это значение в клетку $x_{32}$ и исключаем из рассмотрения второй столбец. На базе $A_3$ остается изменённый запас $a'_3 = 140$.
3. В оставшейся новой таблице с тремя строками $A_1$,$A_2$,$A_3$ и четырьмя столбцами $B_1$,$B_3$,$B_4$,$B_5$ клеткой с наименьшим значением $c_{ij}$ клетка, где $c_{34}$=11. Заполняем описанным выше способом эту клетку и аналогично заполняем следующие клетки. В результате оказываются заполненными (в приведенной последовательности) следующие клетки: $x_{32} = 110, x_{34} = 120, x_{13} = 100, x_{35} = 20, x_{15} = 180, x_{11} = 20, x_{21} = 150$
4. На пятом шаге клеток с наименьшими значениями $c_{ij}$ оказалось две ($c_{11}$=$c_{15}$=70) . Мы заполнили клетку для $x_{15}$ , положив $x_{15}$ = 180. Можно было выбрать для заполнения другую клетку, положив $x_{11}$ = 170, что приведет в результате к другому опорному плану.

5. Общий объем перевозок в тонно-километрах для этого плана составит:

$$S_1 = 70 \cdot 20 + 15 \cdot 100 + 70 \cdot 180 + 80 \cdot 150 + 10 \cdot 110 + 11 \cdot 120 + 25 \cdot 20 = 30420$$

Замечание: В диагональном методе не учитываются величины тарифов, в методе же наименьшей стоимости эти величины учитываются, и часто последний метод приводит к плану с меньшими общими затратами (что и имеет место в нашем примере), хотя это и не обязательно.

Кроме рассмотренных выше способов иногда используется, так называемый, метод Фогеля. Суть его состоит в следующем: В распределительной таблице по строкам и столбцам определяется разность между двумя наименьшими тарифами. Отмечается наибольшая разность. Далее в строке (столбце) с наибольшей разностью заполняется клетка с наименьшим тарифом. Строки (столбцы) с нулевым остатком груза в дальнейшем в расчет не принимаются. На каждом этапе загружается только одна клетка. Распределение груза производится, как и ранее.