Лабораторная работа 4

МЕТОДЫ ПОИСКА В ПРОСТРАНСТВЕ СОСТОЯНИЙ

Цель работы: Ознакомиться с методами поиска решения интеллектуальных задач в пространстве состояний. Изучить блок-схемы алгоритмов для всех методов поиска.

Теоретические сведения

При формулировке задачи в пространстве состояний решение получается в результате применения операторов к описаниям состояний до тех пор, пока не будет получено выражение, описывающее состояние, которое соответствует достижению цели. На примерах, рассмотренных в предыдущей главе, мы видели, как для иллюстрации пространств состояний могут быть использованы графы. Язык графов чрезвычайно полезен для описания эффективных стратегий перебора (поиска) в пространстве состояний. Все методы перебора в пространстве состояний, которые мы будем обсуждать, могут быть смоделированы с помощью следующего теоретико-графового процесса:

Начальная вершина соответствует описанию начального состояния.

Вершины, непосредственно следующие за данной, получаются в результате использования операторов, которые применимы к описанию состояния, ассоциированного с этой вершиной. Пусть Г — некоторый специальный оператор, который строит *все* вершины, непосредственно следующие за данной. Мы будем - называть процесс применения оператора Г к вершине *раскрытием* вершины.

От каждой такой последующей вершины к породившей ее идут *указатели.* Эти указатели позволяют найти путь назад к начальной вершине, уже после того как обнаружена целевая вершина.

Для вершин, следующих за данной, делается проверка, не являются ли они целевыми вершинами. (То есть проверка того, не определяют ли соответствующие описания такие состояния, которые соответствуют цели.) Если целевая вершина еще не найдена, то продолжается процесс раскрытия вершин (и установки указателей). Когда же целевая вершина найдена, эти указатели просматриваются в обратном направлении - от цели к началу, в результате чего выявляется путь решения. Тогда операторы над описаниями состояний, связанные с дугами этого пути, образуют решающую последовательность.

Этапы, указанные выше, описывают просто основные элементы процесса перебора подобно описанию, даваемому блок-схемой недетерминированной программы. При полном описании процесса перебора нужно еще задать порядок, в котором следует раскрывать вершины. Если вершины раскрываются в том же порядке, в котором они порождаются, то получается процесс, который называется *полным перебором* (breadth-first process). Если же сначала раскрывается всегда та вершина, которая была построена самой последней, то получается процесс *перебора в глубину* (depth-first process). Процессы полного перебора и перебора в глубину можно назвать также *процедурами слепого перебора,* поскольку расположение цели не влияет на порядок, в котором раскрываются вершины.

Возможно, однако, что у нас имеется некоторая эвристическая информация о глобальном характере графа и общем расположении цели поиска. (Слово *эвристический* означает «служащий открытию».) Такого рода информация часто может быть использована для того, чтобы «подтолкнуть» поиск в сторону цели, раскрывая в первую очередь наиболее перспективные вершины. В этой главе мы опишем несколько эвристических методов перебора в терминах теории графов. Но прежде мы введем ряд основных идей, связанных с перебором, рассмотрев более подробно методы слепого перебора.

Сущность этих методов станет понятнее, если мы ограничимся рассмотрением деревьев, а не произвольных графов. *Деревом* называется граф, каждая вершина которого имеет ровно одну непосредственно предшествующую ей (родительскую) вершину, за исключением выделенной вершины, называемой *корнем* дерева, которая вовсе не имеет предшествующих ей вершин. Таким образом, корень дерева служит начальной вершиной. Для перебора деревья проще графов прежде всего потому, что при построении новой вершины мы можем быть уверены, что она никогда раньше не строилась и никогда не будет построена вновь. Таким образом, путь от корня до данной вершины единствен. После описания методов слепого поиска для деревьев мы покажем, как их следует модифицировать в случае произвольных графов.

МЕТОДЫ ПОЛНОГО ПЕРЕБОРА

В *методе полного перебора* вершины раскрываются в том порядке, в котором они строятся. Простой алгоритм полного

перебора на дереве состоит из следующей последовательности шагов:

(1) Поместить начальную вершину в список, называемый ОТКРЫТ.

(2) Если список ОТКРЫТ пуст, то на выход подается сигнал о неудаче поиска, в противном случае переходить к следующему шагу.

(3) Взять первую вершину из списка ОТКРЫТ и перенести «ее в список ЗАКРЫТ; назовем эту вершину *n.*

(4) Раскрыть вершину *n,* образовав все вершины, непосредственно следующие за *n.* Если непосредственно следующих вершин нет, то переходить сразу же к шагу (2). Поместить имеющиеся непосредственно следующие за *n* вершины в конец списка ОТКРЫТ и построить указатели, ведущие от них назад к вершине *n*.

(5) Если какие-нибудь из этих непосредственно следующих за *n* вершин являются целевыми вершинами, то на выход выдать решение, получающееся просмотром вдоль указателей; в противном случае переходить к шагу (2).

В этом алгоритме предполагается, что начальная вершина не удовлетворяет поставленной цели, хотя нетрудно ввести этап проверки такой возможности. Блок-схема алгоритма показана на рис. 3.1. Вершины и указатели, построенные в процессе перебора, образуют поддерево всего неявно определенного дерева пространства состояний. Мы будем называть такое поддерево *деревом перебора.*

Легко показать, что в методе полного перебора непременно будет найден самый короткий путь к целевой вершине при условии, что такой путь вообще существует. (Если такого пути нет, то в указанном методе будет объявлено о неуспехе в случае конечных графов, а в случае бесконечных графов алгоритм никогда не кончит свою работу.)

На рис. 3.2 приведено дерево перебора, полученное в результате полного перебора, примененного к игре в восемь. (Граф пространства состояний для игры в восемь в действительности деревом не является, но этот факт несуществен, так как в рассматриваемом процессе перебора одна и та же вершина никогда не может возникнуть более, чем от одной родительской вершины.) Задача состоит в том, чтобы преобразовать конфигурацию, стоящую слева, в конфигурацию, стоящую справа:

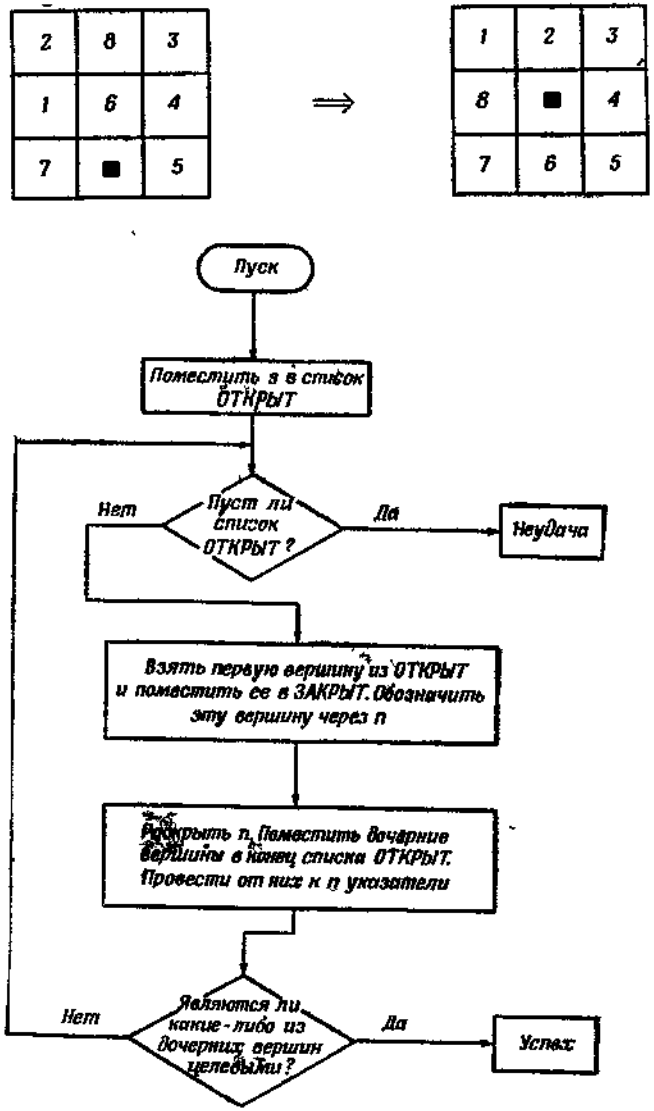


Рис. 3.1. Блок-схема программы алгоритма полного перебора для деревьев.

В вершинах дерева помещены соответствующие описания состояний. Эти вершины занумерованы в том порядке, в котором они получались при раскрытии (порядок последующих вершин соответствует перемещению пустой клетки сначала влево, затем вверх, вправо и вниз); зачерненная ветвь представляет собой решение из пяти шагов. (Стрелки на дугах не указаны, поскольку в данном случае совершенно ясно происхождение каждой вершины.)

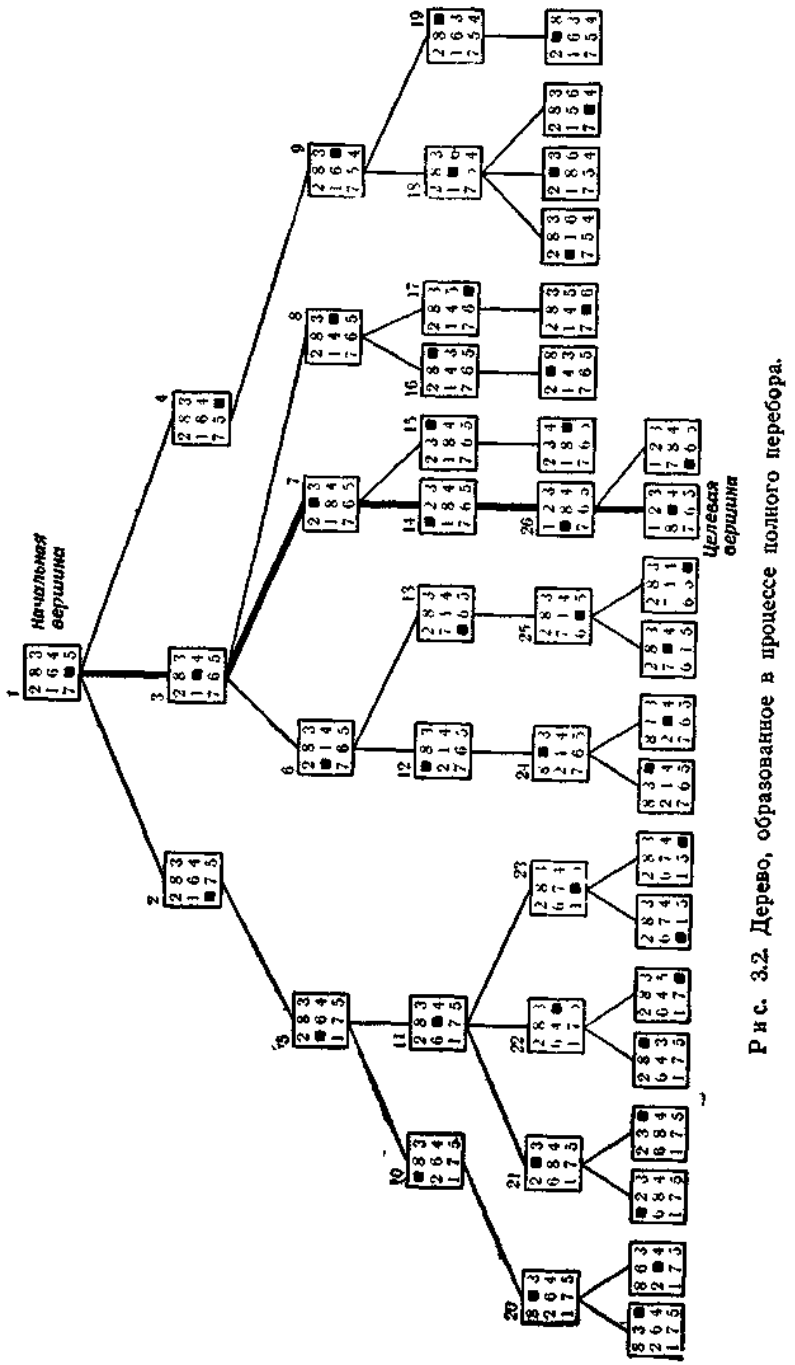


Рис. 3.2. Дерево, образованное в процессе полного перебора.

Заметьте, что было раскрыто 26 и построено 46 вершин, прежде чем удалось найти это решение. Непосредственное рассмотрение этого, графа показывает также, что не существует решения, содержащего меньшее число шагов.

Могут встретиться задачи, в которых к решению предъявляются какие-то иные требования, отличные от требования получения наикратчайшей последовательности операторов. Присваивание дугам дерева определенных цен (с последующим нахождением решающего пути, имеющего минимальную стоимость) соответствует многим из таких обобщенных критериев, как это было видно из нескольких примеров предыдущей главы. Более общий вариант метода полного перебора, называемый ***методом равных цен***, позволяет во всех случаях найти некоторый путь от начальной вершины к целевой, стоимость которого минимальна. В то время как в только что описанном алгоритме распространяются линии равной длины пути от начальной вершины, в более общем алгоритме, который будет описан ниже, распространяются линии равной стоимости пути. Предполагается, что нам задана функция стоимости $c(n_i, n_j)$, дающая стоимость перехода от вершины n_i к некоторой следующей за ней вершине n_j.

В методе равных цен для каждой вершины n в дереве перебора нам нужно помнить стоимость пути, построенного от начальной вершины s к вершине n. Пусть $\hat{g}(n)$—стоимость пути от вершины s к вершине n в дереве перебора. В случае деревьев перебора мы можем быть уверены, что $\hat{g}(n)$ является к тому же стоимостью того пути, который имеет *минимальную* стоимость (так как этот путь единственный).

В ***методе равных цен*** вершины раскрываются в порядке возрастания стоимости $\hat{g}(n)$. Этот метод характеризуется такой последовательностью шагов:

(1) Поместить начальную вершину s в список, называемый ОТКРЫТ. Положить $\hat{g}(s) = 0$.

(2) Если список ОТКРЫТ пуст, то на выход подается сигнал о неудаче поиска; в противном случае переходить к следующему шагу.

(3) Взять из списка ОТКРЫТ ту вершину, для которой величина $\hat{g}$ имеет наименьшее значение, и поместить ее в список, называемый ЗАКРЫТ. Дать этой вершине название n. (В случае совпадения значений выбирать вершину с минимальным $\hat{g}$ произвольно, но всегда отдавая предпочтение целевой вершине.)

(4) Если *n* есть целевая вершина, то на выход выдать решающий путь, получаемый путем просмотра назад в соответствии с указателями; в противном случае переходить к следующему шагу.

(5) Раскрыть вершину *n,* построив все непосредственно следующие за ней вершины. [Если таковых не оказалось, то переходить сразу к шагу (2).] Для каждой из такой непосредственно следующей (дочерней) вершины n_i вычислить стоимость $\hat{g}(n_i)$, положив $\hat{g}(n_i) = \hat{g}(n) + c(n,n_i)$. Поместить эти вершины вместе с соответствующими им только что найденными значениями $\hat{g}$ в список ОТКРЫТ и построить указатели, идущие назад к *n.*

(6) Перейти к шагу (2).

Блок-схема этого алгоритма показана на рис. 3.3**.**

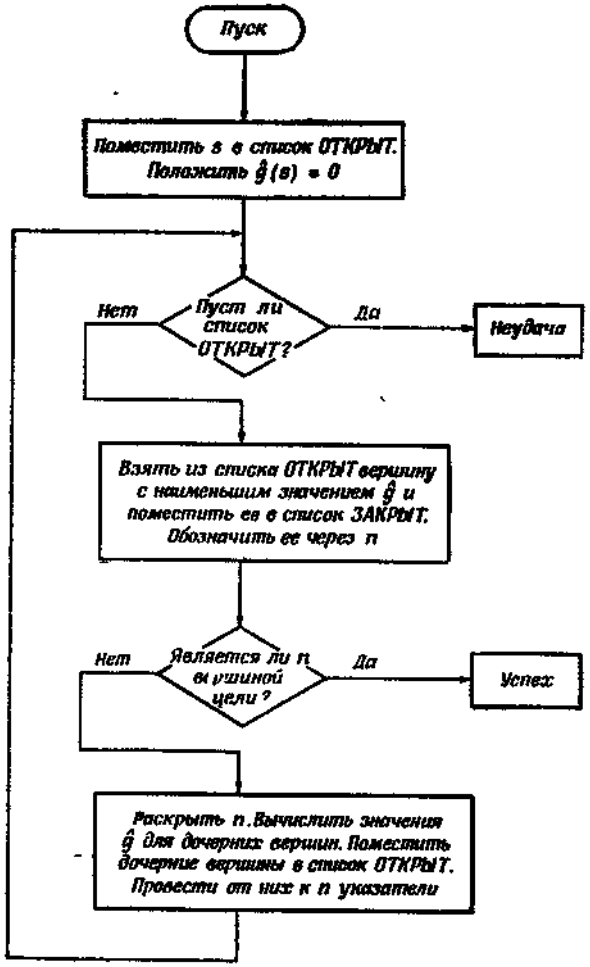


Рис. 3.3. Блок-схема программы алгоритма равных цен для деревьев.

Заметьте, что проверка того, является ли некоторая вершина целевой, включена в эту схему так, что гарантируется обнаружение путей минимальной стоимости.

Мы видим, что алгоритм, работающий по методу равных цен, может быть также использован для поиска путей минимальной длины, если просто положить стоимость каждого ребра равной *единице.* Если имеется несколько начальных вершин, то алгоритм просто модифицируется: на шаге (1) *все* начальные вершины помещаются в список ОТКРЫТ. Если состояния отвечающие поставленной цели, могут быть описаны явно, то процесс перебора можно пустить в обратном направлении, приняв целевые вершины в качестве начальных и используя обращение оператора Г.

МЕТОД ПЕРЕБОРА В ГЛУБИНУ

В *методах перебора в глубину* прежде всего раскрываются те вершины, которые были построены последними. Определим *глубину* вершины в дереве следующим образом:

Глубина корня дерева равна нулю.

Глубина любой последующей вершины равна единице плюс глубина вершины, которая непосредственно ей предшествует.

Таким образом, вершиной, имеющей наибольшую глубину в дереве перебора, в данный момент служит та, которая должна в этот момент быть раскрыта. Такой подход может привести к процессу, разворачивающемуся вдоль некоторого бесполезного пути, поэтому нужно ввести некоторую процедуру возвращения. После того как в ходе процесса строится вершина с глубиной, превышающей некоторую *граничную глубину,* раскрывается вершина наибольшей глубины, не превышающей этой границы, и т.д.

Метод перебора в глубину определяется следующей последовательностью шагов:

(1) Поместить начальную вершину в список, называемый ОТКРЫТ.

(2) Если список ОТКРЫТ пуст, то на выход дается сигнал о неудаче поиска, в противном случае перейти к шагу (3).

(3) Взять *первую* вершину из списка ОТКРЫТ и перенести ее в список, называемый ЗАКРЫТ. Эту вершину назвать *n.*

(4) Если глубина вершины *n* равна граничной глубине, то переходить к (2), в противном случае - к (5).

(5) Раскрыть вершину *n,* построив все непосредственно следующие за ней вершины. Поместить их (в произвольном порядке) в *начало* списка ОТКРЫТ и построить указатели, идущие от них к вершине *n.*

(6) Если одна из этих вершин целевая, то на выход выдать решение, просматривая для этого соответствующие указатели, в противном случае переходить к шагу (2).

На рис.3.4. приведена блок-схема для метода перебора в глубину.

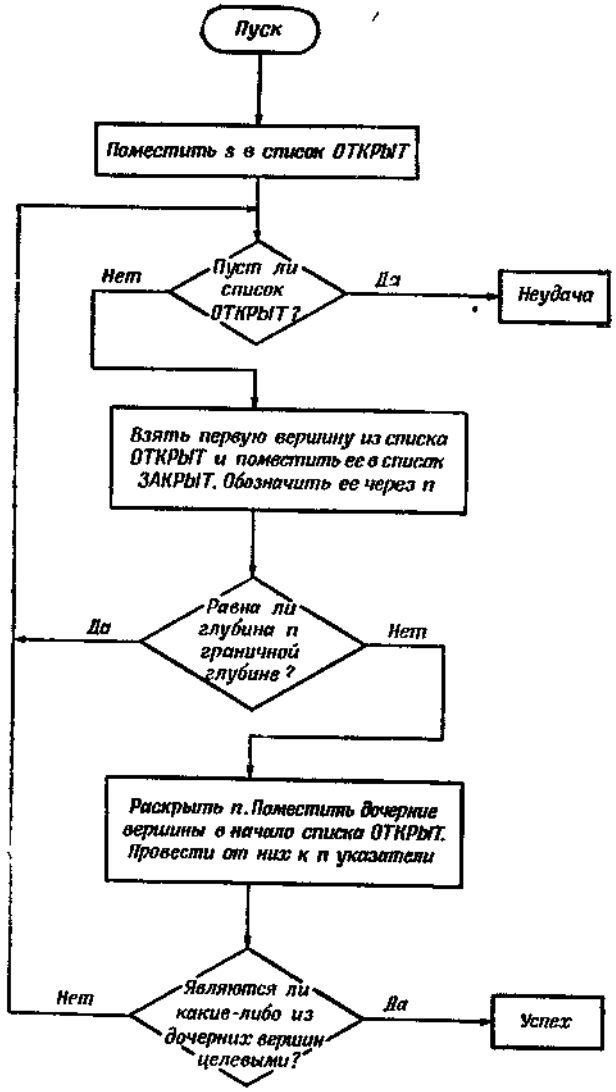


Рис. 3.4. Блок-схема программы алгоритма поиска в глубину для деревьев.

Дерево, которое получается в результате применения метода перебора в глубину к той же самой игре в восемь, что и прежде, показано на рис. 3.5.

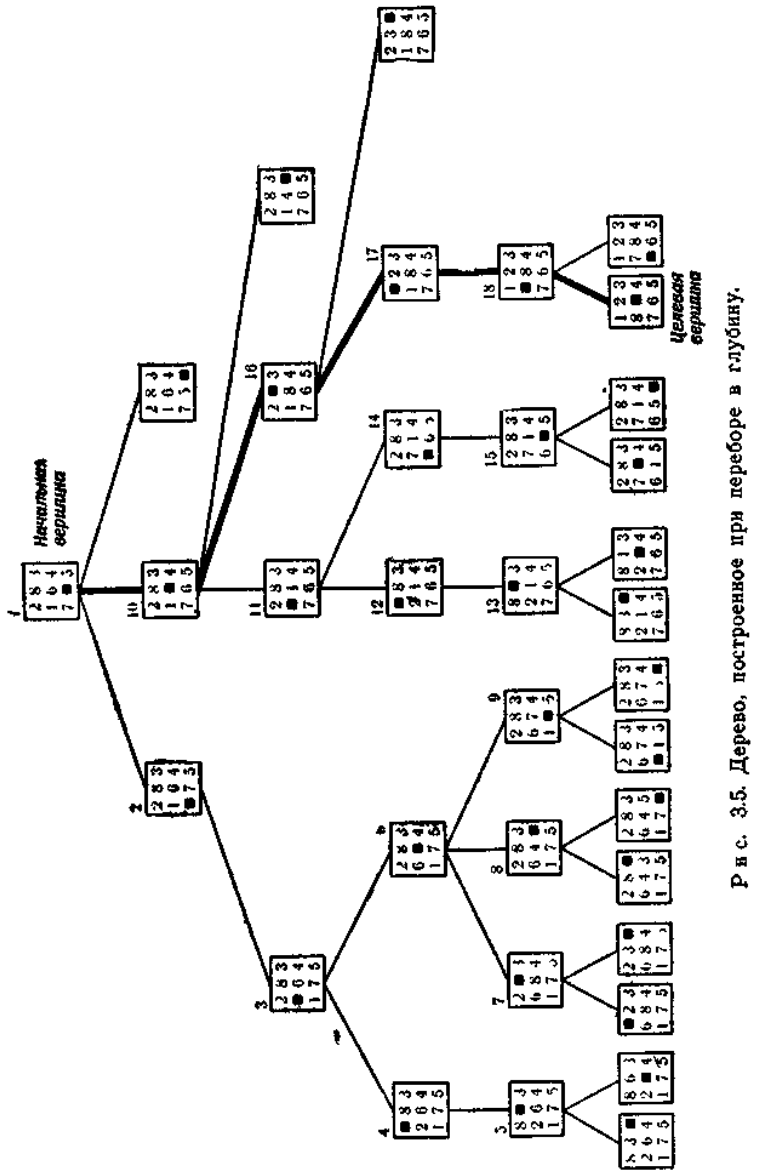
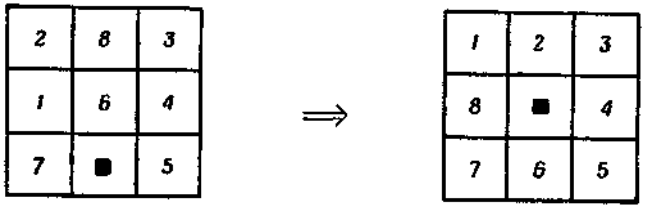


Рис. 3.5. Дерево, построенное при переборе в глубину.

Снова нужно найти последовательность ходов для преобразования левой конфигурации в правую:



Вершины здесь занумерованы в том порядке, в котором они были раскрыты, причем граничная глубина выбрана равной 5 (т.е. ищутся пути, ведущие к цели, длина которых не больше пяти).

Видно, что в алгоритме поиска в глубину сначала идет перебор вдоль одного пути, пока не будет достигнута максимальная глубина, затем рассматриваются альтернативные пути той же или меньшей глубины, которые отличаются от него лишь последним шагом, после чего рассматриваются пути, отличающиеся последними двумя шагами, и т.д.

ИЗМЕНЕНИЯ ПРИ ПЕРЕБОРЕ НА ПРОИЗВОЛЬНЫХ ГРАФАХ

При переборе на графах, а не на деревьях, нужно внести некоторые естественные изменения в указанные алгоритмы. В простом методе полного перебора не нужно вносить никаких изменений; следует лишь проверять, не находится ли уже вновь построенная вершина в списках ОТКРЫТ или ЗАКРЫТ по той причине, что она уже строилась раньше в результате раскрытия какой-то вершины. Если это так, то ее не нужно вновь помещать в список ОТКРЫТ.

Несколько более сложные изменения должны быть сделаны в алгоритме равных цен:

(1) Если вновь построенная вершина уже имеется в списке ОТКРЫТ, то ее не следует вносить в этот список снова. Однако соответствующая ей величина стоимости $\hat{g}$ может оказаться теперь меньше (может быть найден менее дорогой путь). Мы всегда связываем с вершинами списка ОТКРЫТ наименьшие из имевшихся до сих пор значений $\hat{g}$. Точно так же указатель от вершины всегда должен быть направлен к породившей ее вершине, расположенной на том пути, стоимость которого оказалась наименьшей среди всех путей к этой вершине, рассмотренных к настоящему моменту.

(2) Если вновь построенная вершина уже имеется в списке ЗАКРЫТ, то, казалось бы, возможно, что для нее величина $\hat{g}$ окажется меньше, чем раньше, так как направление ее указателя должно быть выбрано заново. Но на самом деле этого не происходит. Позже мы докажем, что если в алгоритме равных цен некоторая вершина помещается в список ЗАКРЫТ, то уже найдена наименьшая возможная величина $\hat{g}$ (и, следовательно, путь наименьшей стоимости, идущий к этой вершине).

Прежде чем делать какие-либо изменения в алгоритме перебора в глубину, нужно решить, что понимать под *глубиной* вершины в графе. Согласно обычному определению, глубина вершины равна единице плюс глубина наиболее близкой родительской вершины, причем глубина начальной вершины предполагается равной нулю. Тогда поиск в глубину можно было бы получить, выбирая для раскрытия самую глубокую вершину списка ОТКРЫТ (без превышения граничной глубины) Когда порождаются вершины, уже имеющиеся либо в списке ОТКРЫТ, либо в списке ЗАКРЫТ, пересчет глубины такой вершины может оказаться необходимым.

Даже в том случае, когда перебор осуществляется на полном графе, множество вершин и указателей, построенное в процессе перебора, тем не менее образует *дерево*. (Указатели по-прежнему указывают только на одну порождающую вершину.) В оставшейся части этой главы мы имеем дело с общим случаем поиска на графе, и, следовательно, в алгоритмах, которые мы будем обсуждать, явным образом учитываются эти дополнительные изменения.

ОБСУЖДЕНИЕ ЭВРИСТИЧЕСКОЙ ИНФОРМАЦИИ

Методы слепого перебора, полного перебора или поиска в глубину являются исчерпывающими процедурами поиска путей к целевой вершине. В принципе эти методы обеспечивают решение задачи поиска пути, но часто эти методы невозможно использовать, поскольку при переборе придется раскрыть слишком много вершин, прежде чем нужный путь будет найден. Так как всегда имеются практические ограничения на время вычисления и объем памяти, отведенные для перебора, то мы должны заняться поисками других методов, более эффективных, чем методы слепого перебора.

Для многих задач можно сформулировать чисто эмпирические правила, позволяющие уменьшить объем перебора. Все такие

правила, используемые для ускорения поиска, зависят от специфической информации о задаче, представляемой в виде графа. Будем называть информацию такого сорта *эвристической информацией* (помогающей найти решение) и называть использующие ее процедуры поиска *эвристическими методами поиска.* Один из путей уменьшить перебор состоит в выборе более «информированного», оператора Г, который не строит так много не относящихся к делу вершин. Этот способ применим как в методе полного перебора, так и в методе перебора в глубину. Другой путь состоит в использовании эвристической информации для модификации шага (5) алгоритма перебора в глубину. Вместо того чтобы размещать вновь построенные вершины в произвольном порядке в начале списка ОТКРЫТ, их можно расположить в нем некоторым определенным образом, зависящим от эвристической информации. Так, при переборе в глубину в первую очередь будет раскрываться та вершина, которая представляется наилучшей.

Более гибкий (и более дорогой) путь использования эвристической информации состоит в том, чтобы, согласно некоторому критерию, на каждом шаге переупорядочивать вершины списка ОТКРЫТ. В этом случае перебор мог бы идти дальше в тех участках границы, которые представляются наиболее "перспективными. Для того чтобы применять процедуру упорядочения, нам необходима мера, которая позволяла бы оценивать «перспективность» вершины. Такие меры называют *оценочными функциями.*

Как мы сможем убедиться, подчас удается выделить эвристическую информацию (эвристику), уменьшающую усилия, затрачиваемые на перебор (до величины, меньшей, скажем, чем при поиске методом равных цен), без потери гарантированной возможности найти путь, обладающий наименьшей стоимостью. Чаще же используемые эвристики сильно уменьшают объем работы, связанной с перебором, ценою отказа от гарантии найти путь наименьшей стоимости в некоторых или во всех задачах. Но в большинстве практических задач мы заинтересованы в минимизации некоторой *комбинации* из стоимости пути и стоимости перебора, необходимого для нахождения этого пути.

Более того, нас обычно интересуют методы перебора, при которых минимизируется такая комбинация, *усредненная* по всем задачам, которые могут нам встретиться. (При вычислении этой средней стоимости мы должны использовать веса, пропорциональные частоте появления каждой задачи, так что мы допускаем большие стоимости для нечасто встречающихся задач, если они компенсируются меньшими стоимостями для часто встречающихся задач.) Если усредненная комбинационная стоимость одного метода перебора ниже соответствующей стоимости другого метода, то мы говорим, что первый метод перебора обладает большей *эвристической силой,* чем второй. Заметим, что по нашему определению вовсе не обязательно (хотя это обычно недопонимается), чтобы метод перебора, имеющий большую эвристическую силу, чем другой, гарантировал возможность нахождения путей минимальной стоимости, которую другой метод мог бы обеспечивать.

Усредненные комбинационные стоимости в действительности никогда не вычисляются как по той причине, что трудно выбрать способ комбинирования стоимости пути и стоимости усилий, затрачиваемых на перебор, так и потому, что было бы нелегко найти вероятностное распределение на множестве задач, которые могут встретиться. Таким образом, решение вопроса о том, имеет ли один метод перебора большую эвристическую силу по сравнению с другим, зиждется обычно на интуиции знатока, подкрепленной реальными экспериментами с этими методами.

ИСПОЛЬЗОВАНИЕ ОЦЕНОЧНЫХ ФУНКЦИЙ

Как мы уже отмечали, обычный способ использования эвристической информации связан с употреблением для упорядочения перебора *оценочных функций.* Оценочная функция должна обеспечивать возможность ранжирования вершин - кандидатов на раскрытие - с тем, чтобы выделить ту вершину, которая с наибольшей вероятностью находится на лучшем пути к цели. Оценочные функции строились на основе различных соображений. Делались попытки определить *вероятность* того, что вершина расположена на лучшем пути. Предлагалось также использовать *расстояние* или другие меры *различия* между произвольной вершиной и множеством целевых вершин. В салонных играх или головоломках позиции часто ставится в соответствие определенное число очков на основе тех черт, которыми она обладает и которые представляются связанными со степенью уверенности в том, что это шаг к поставленной цели.

Предположим, что задана некоторая функция f, которая могла бы быть использована для упорядочения вершин перед их

раскрытием. Через $\hat{f}(n)$ обозначим значение этой функции на вершине n. Пока мы будем считать, что $\hat{f}$ — произвольная функция. Позднее же мы предположим, что она совпадает с оценкой стоимости того из путей, идущих от начальной вершины к целевой и проходящих через вершину n, стоимость которого — наименьшая (из всех таких путей).

Условимся располагать вершины, предназначенные для раскрытия, в порядке *возрастания* их значений функции $\hat{f}$. Тогда можно использовать некоторый алгоритм (подобный алгоритму равных цен), в котором для очередного раскрытия выбирается та вершина списка ОТКРЫТ, для которой значение $\hat{f}$ оказывается наименьшим. Будем называть такую процедуру *алгоритмом упорядоченного перебора* (ordered-search algorithm).

Чтобы этот алгоритм упорядоченного перебора был применим для перебора на произвольных графах (а не только на деревьях), необходимо предусмотреть в нем возможность работы в случае построения вершин, которые уже имеются либо в списке ОТКРЫТ, либо в списке ЗАКРЫТ. При использовании некоторой произвольной функции $\hat{f}$ нужно учесть, что величина $\hat{f}$ для некоторой вершины из списка ЗАКРЫТ может понизиться, если к ней найден новый путь ($\hat{f}(n)$ может зависеть от пути из s к n даже для вершин списка ЗАКРЫТ). Следовательно, мы должны тогда перенести такие вершины назад в список ОТКРЫТ и позаботиться об изменении направлений соответствующих указателей.

После принятия этих необходимых мер" алгоритм упорядоченного поиска может быть представлен такой последовательностью шагов:

(1) Поместить начальную вершину S в список, называемый ОТКРЫТ, и вычислить $\hat{f}(s)$.

(2) Если список ОТКРЫТ пуст, то на выход дается сигнал о неудаче; в противном случае переходить к следующему этапу. (3) Взять из списка ОТКРЫТ ту вершину, для которой $\hat{f}$ имеет наименьшее значение, и "поместить ее в список, называемый ЗАКРЫТ. Дать этой вершине имя n. (В случае совпадения значений $\hat{f}$ для нескольких вершин выбирать вершину с минимальным $\hat{f}$ произвольно, но всегда отдавая предпочтение целевой вершине.)

(4) Если n - целевая вершина, то на выход подать решающий путь, получаемый прослеживанием соответствующих указателей; в противном случае переходить к следующему этапу.

(5) Раскрыть вершину n, построив все непосредственно следующие за ней вершины. (Если таковых не оказалось, переходить сразу к (2).) Для каждой такой дочерней вершины n_i вычислить значение $\hat{f}(n_i)$.

(6) Связать с теми из вершин n_i, которых еще нет в списках ОТКРЫТ или ЗАКРЫТ, только что подсчитанные значения $\hat{f}(n_i)$. Поместить эти вершины в список ОТКРЫТ и провести от них к вершине n указатели.

(7) Связать с теми из непосредственно следующих за n вершинами, которые уже были в списках ОТКРЫТ или ЗАКРЫТ, меньшее из прежних и только что вычисленных значений $\hat{f}$ Поместить в список ОТКРЫТ те из непосредственно следующих за n вершин, для которых новое значение $\hat{f}$ оказалось ниже, и изменить направление указателей от всех вершин, для которых значение $\hat{f}$ уменьшилось, направив их к n.

(8) Перейти к (2).

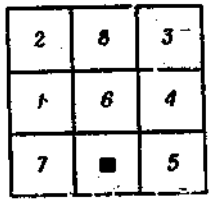
Общая структура алгоритма идентична структуре алгоритма равных цен (см. рис. 3.3), поэтому мы не приводим для него блок-схему. Отметим, что множество, вершин и указателей, порождаемых этим алгоритмом, образует дерево (дерево перебора), причем на концах этого дерева расположены вершины из списка ОТКРЫТ.

Работу алгоритма проще всего пояснить, рассмотрев вновь тот же самый, пример игры в восемь. Мы будем пользоваться следующей простой оценочной функцией:

$$\hat{f}(n)=\hat{g}(n)+W(n),$$

где $\hat{g}(n)$ —длина пути в дереве перебора от начальной вершины до вершины n, а $W(n)$ —это число фишек, которые лежат не на своем месте в описании состояния, связанного с вершиной n.

Так, для начальной вершины



значение $\hat{f}$ равно 0 + 4 = 4. По предположению с большей вероятностью на оптимальном пути находится та вершина, которая имеет наименьшую оценку.

На рис. 3.6 показан результат применения к игре в восемь алгоритма упорядоченного перебора и использования такой оценочной функции. Значение $\hat{f}$ для каждой вершины приведено внутри кружка. Отдельно стоящие цифры показывают порядок, в котором раскрывались вершины. Найден тот же самый путь решений, который был получен другими методами перебора, но

использование оценочной функции привело к существенно меньшему числу раскрытий вершин.

Выбор оценочной функции сильно влияет на результат перебора. Использование оценочной функции, не учитывающей истинной перспективности некоторых вершин, может привести к построению путей, не обладающих минимальной стоимостью. С другой стороны, использование оценочной функции, которая придает слишком большое значение возможной перспективности всех вершин (такой, как в алгоритме равных цен), приведет к тому, что придется раскрыть очень много вершин. В следующих нескольких разделах будет получен ряд теоретических результатов, относящихся к некоторой специальной оценочной функции, использование которой приводит к раскрытию наименьшего числа вершин, совместимого с гарантией нахождения пути минимальной стоимости.

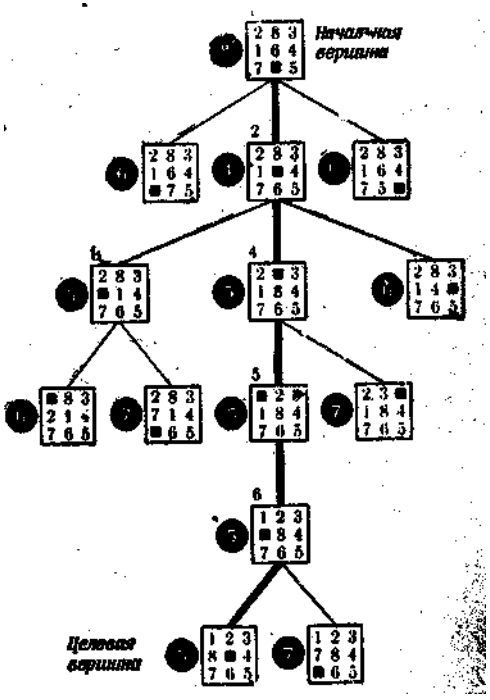


Рис. 3.6. Дерево, построенное в процессе упорядоченного перебора

Завдання до лабораторної роботи № 4/1: Виконати задачу про комівояжера, отримавши свій варіант завдання у викладача **методом пошуку вшир.**

1) Намалювати дерево за методом пошуку вшир.
2) Скласти блок-схему програми алгоритму пошуку рішення;
3) Написати програму;
4) Проаналізувати отриманий результат в лабораторній роботі № 3.

Завдання до лабораторної роботи № 4/2: Виконати задачу про комівояжера, вибравши один з розглянутих методів пошуку.

1) Намалювати дерево за обраним методом;
2) Скласти блок-схему програми алгоритму пошуку рішення за обраним методом;
3) Написати програму;
4) Проаналізувати отриманий результат в лабораторній роботі № 3та № 4/1 зробити висновок.