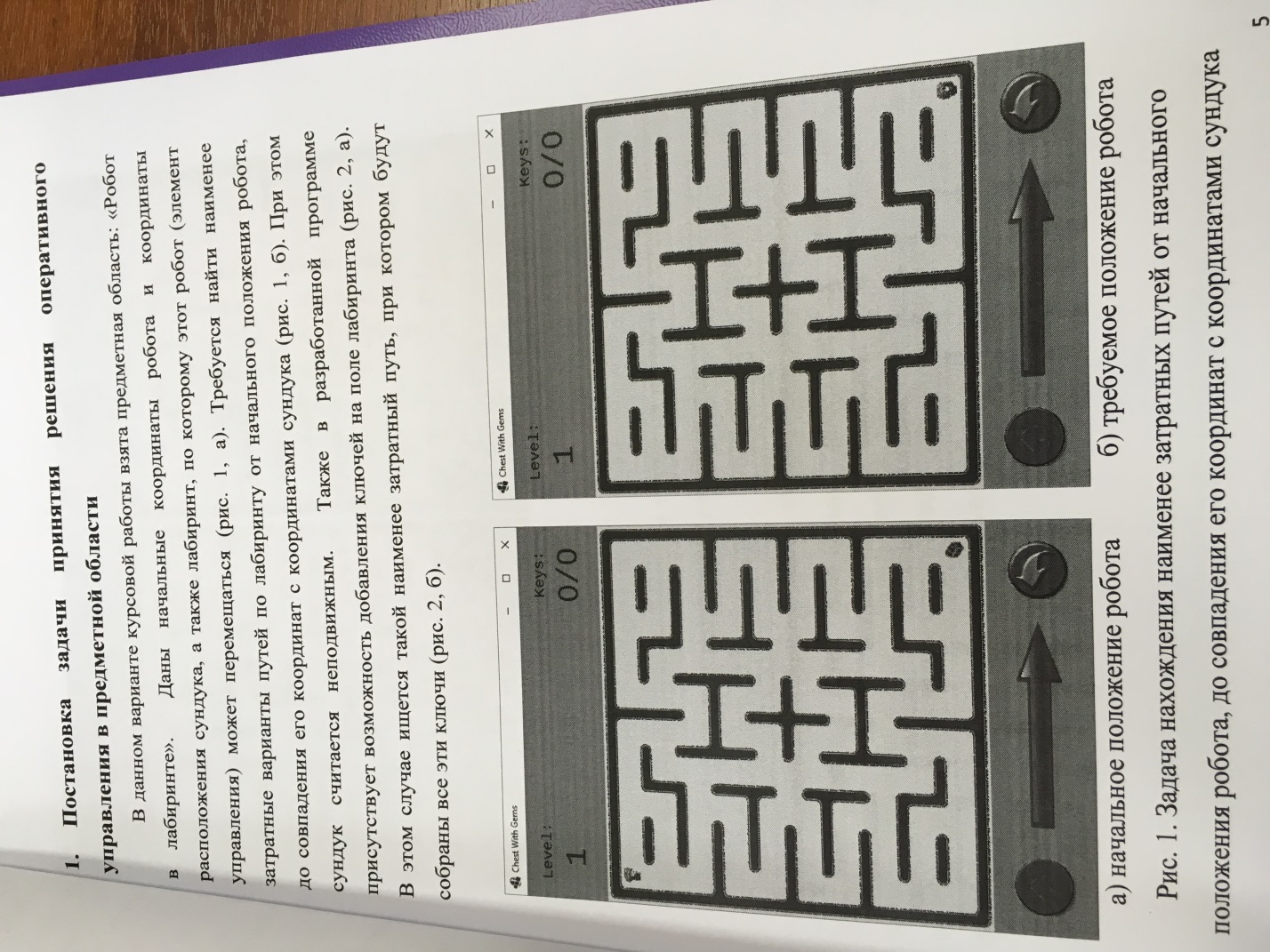
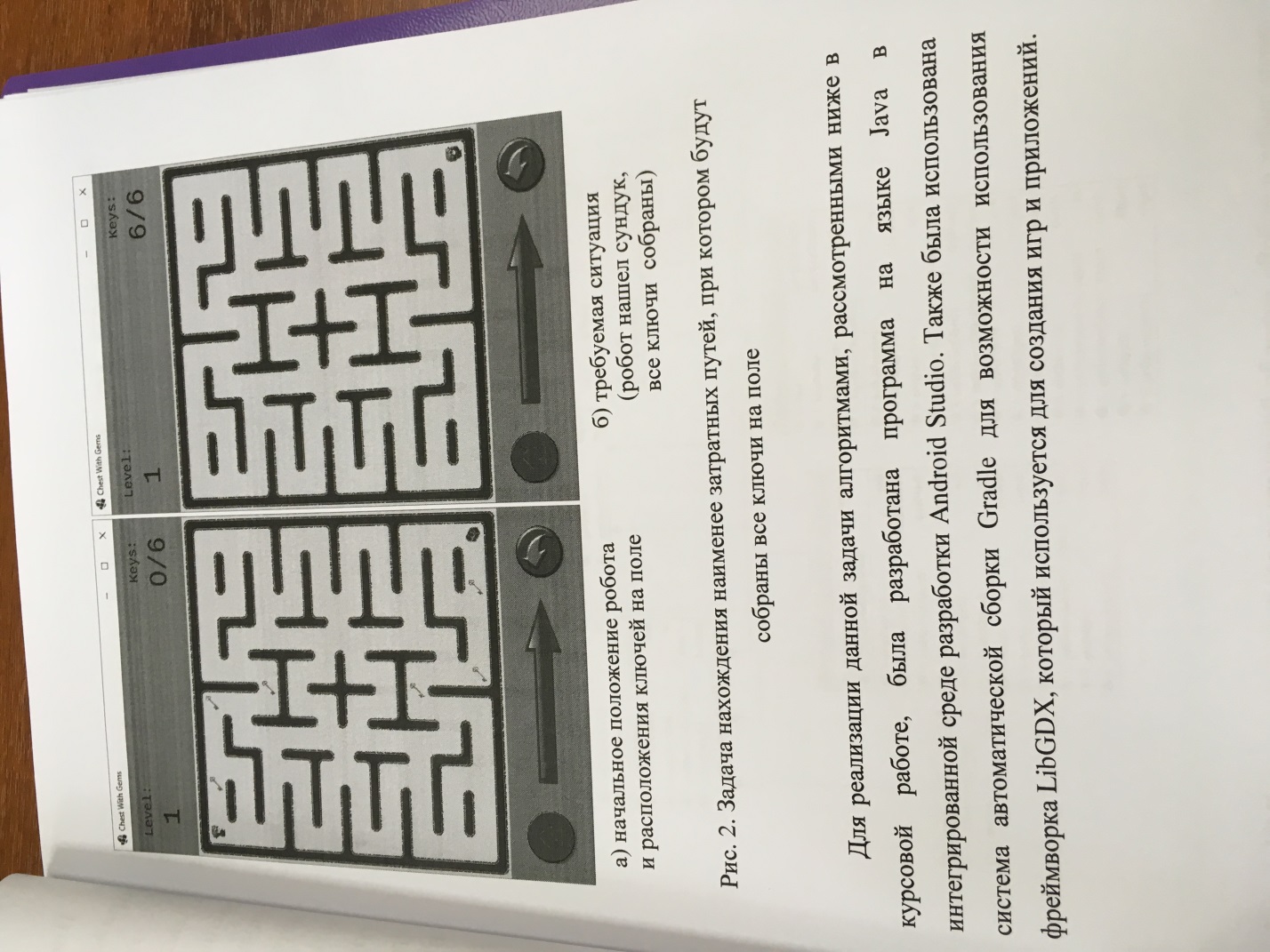
Содержание

# Введение

*объяснить актуальность зачем оперативное управление и зачем нужны интеллектуальные решатели. Где это можно использовать (1 – 3 стр).*

# Постановка задачи принятия решения оперативного управления в предметной области



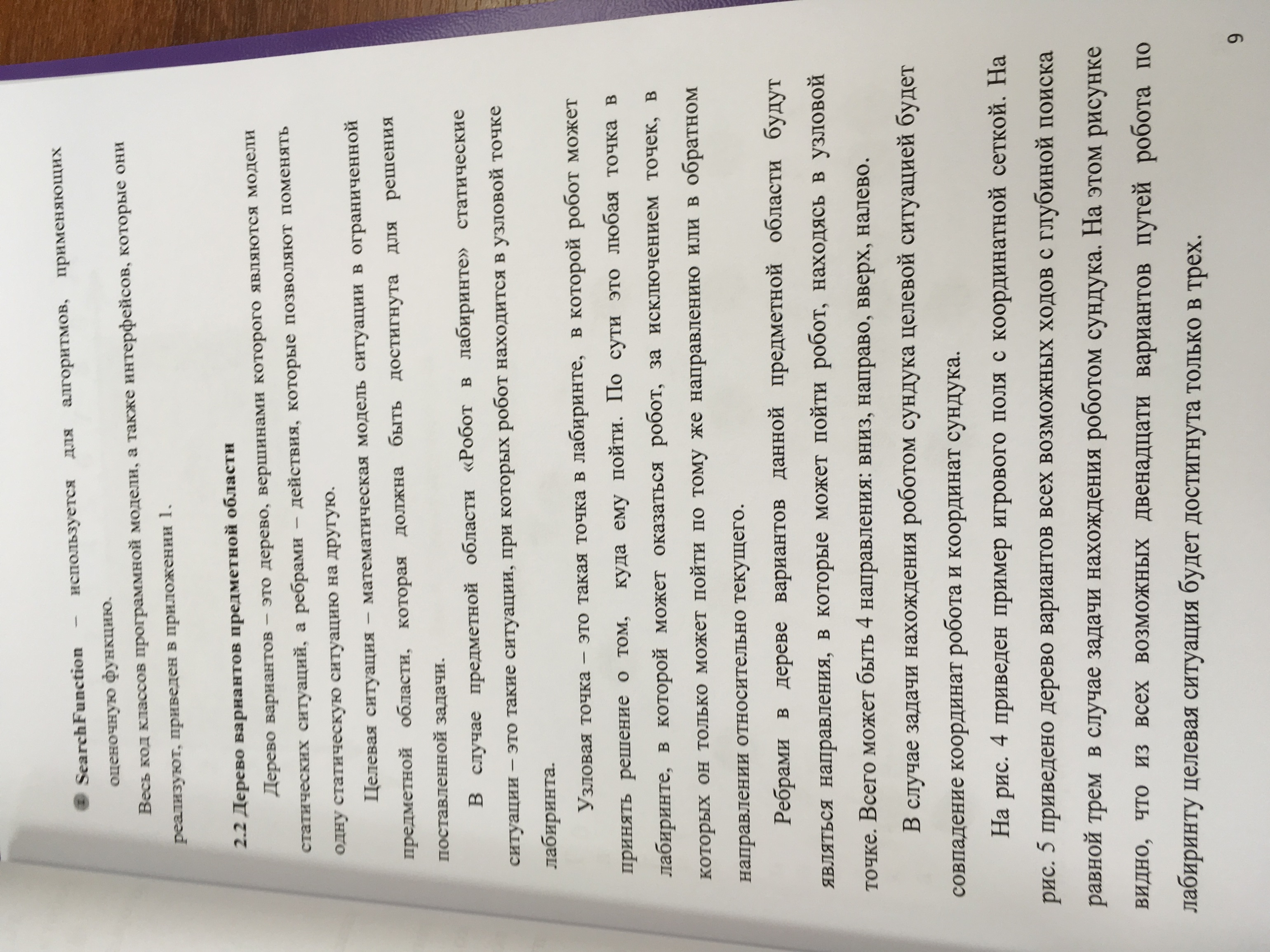
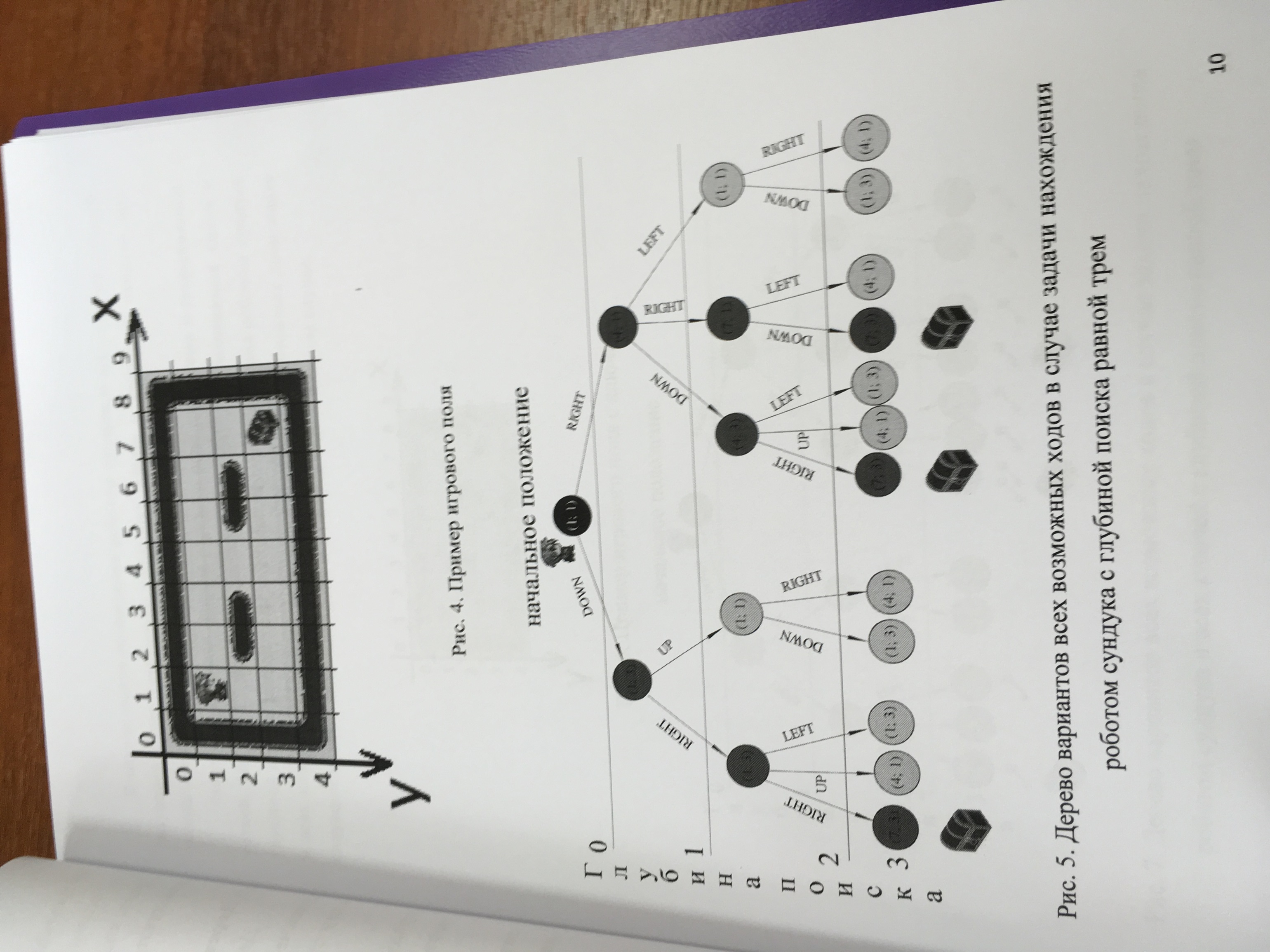
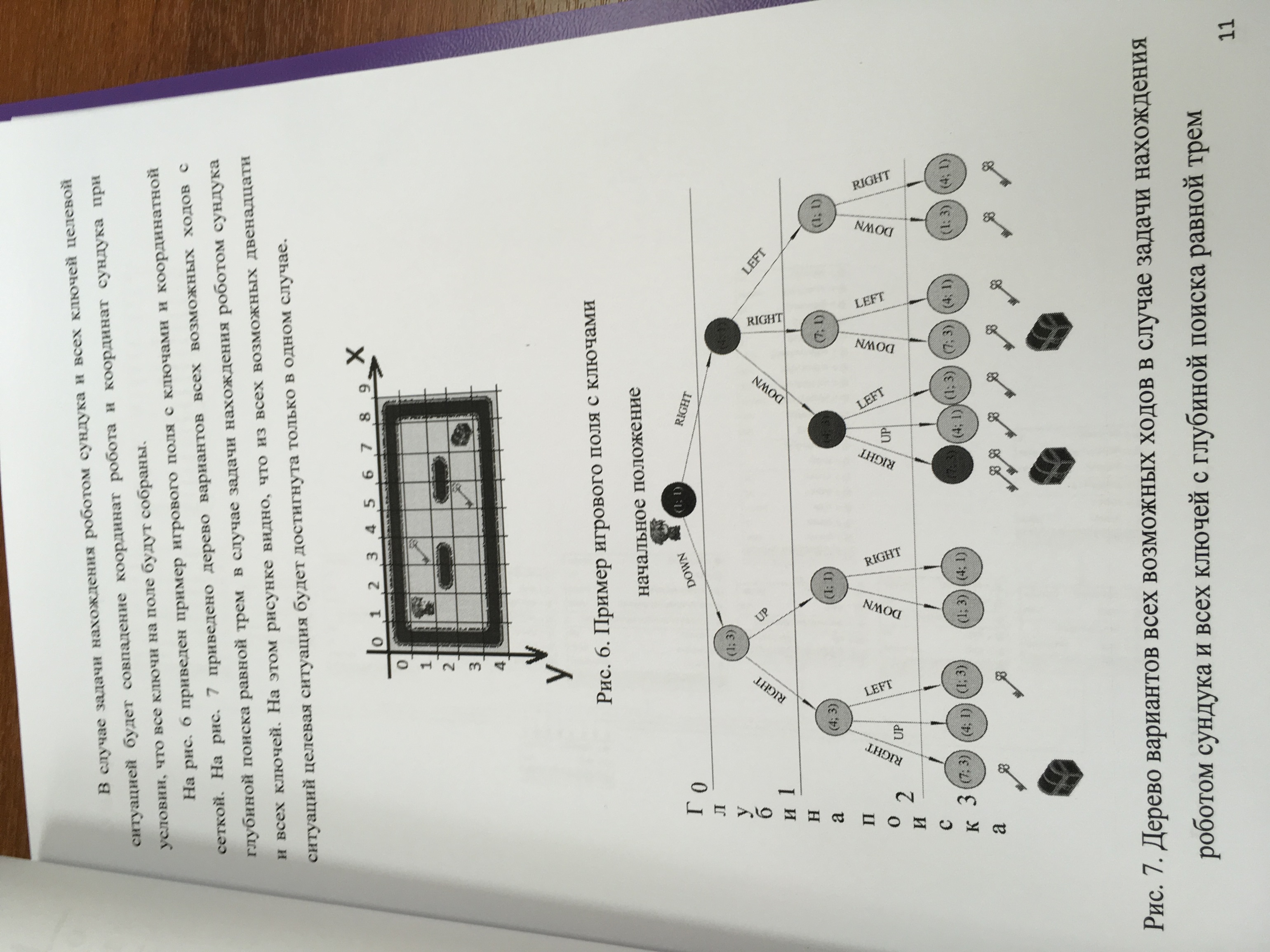
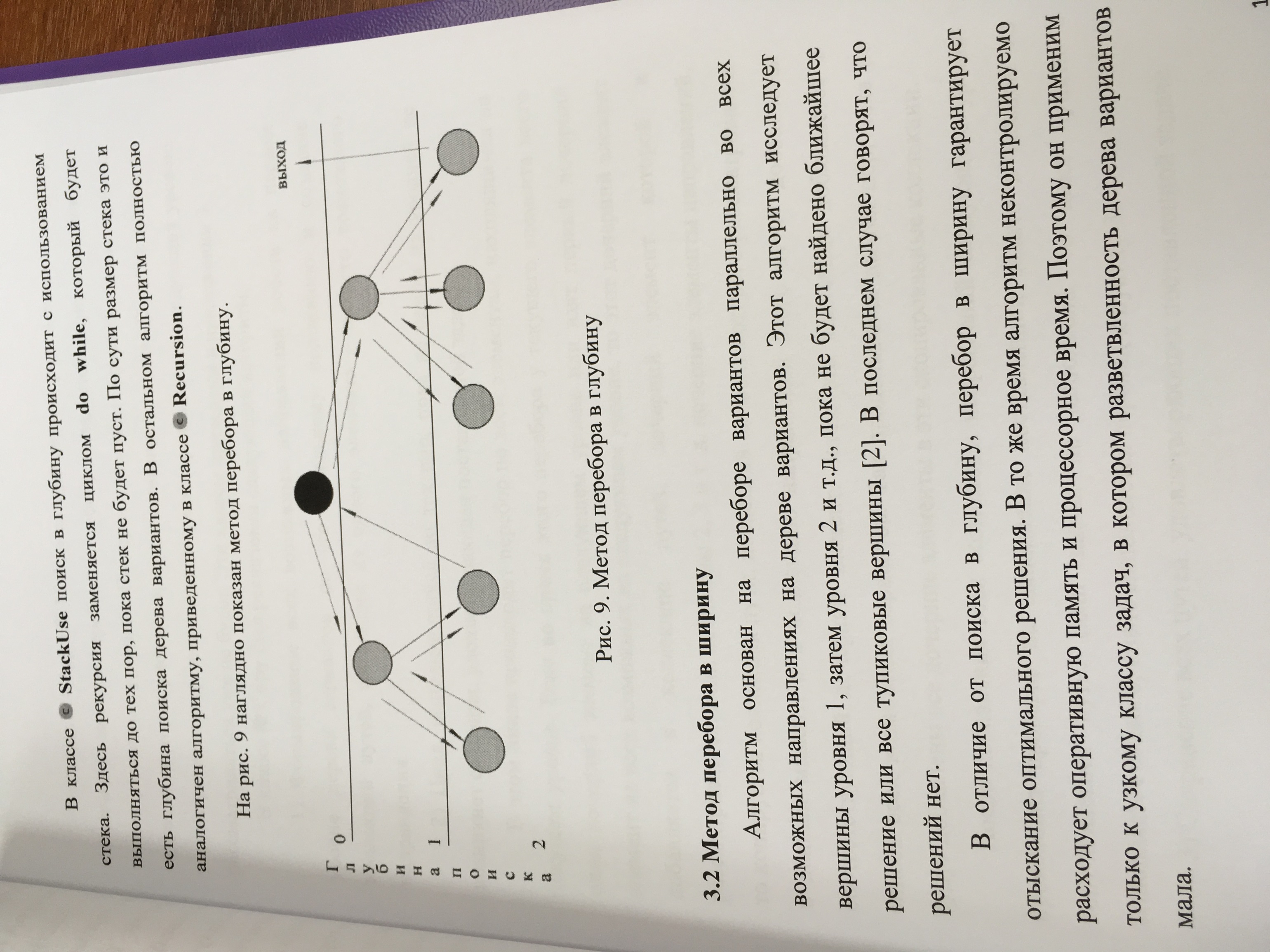
*в чем заключается зад, особенности, что на входе, что на выходе. в задачу входит научное исследование – сравнение различных методов решения задачи*

# Основные классы, их иерархия при проектировании системы принятия решения т.е. структурограмма классов

<https://github.com/ElenaRzn/nine-game>

## 2.1 Программная модель предметной области.

## 2.2 Дерево вариантов предметной области. Описать на каких принципах его строите – список списков, двусвязный список и т.д.



## Структурограмма классов алгоритмов

## опционально – основные методы классов (предикат определения цели, предикат определение тупика (сравнение с глубинй просмотра), метод порождения дочерней вершины.

# 3 Проектирование базовых методов принятия решения

## Метод перебора в глубину (графич. материал – бл. схемы)

## 3.2 Метод перебора в ширину

# 4 Проектирование интеллектуального решателя с использованием предварительных знаний эксперта

## Оценочная ф-я (вообще и в частности).

## Поиск по градиенту

## Стратегия ветвей и границ

## Равных цен

# 5. Сравнение алгоритмов

Для сравнения методов алгоритмов эвристического поиска с целью выявления более эффективных из них по затратам основных компьют. ресурсов рассчитывают характеристики поиска, при этом исследуется составляющаа: 1) время поиска 2) объем занятой оперативной памяти 3) активизированное поддерево вариантов.

Активизированное поддерево – поддерево построенное программно в процессе анализа вариантов решения.

Дерево – это все вершины (не путь) не смотря на то, что большинство вершин не хранили. Акт дерево должно включать все вершины, когда либо построенные программой даже если они не были сохранены в оперативной памяти.

Характеристики поиска разделяют на базовые, произвольные и эффективные. **Базовые** – получаются непосредственно, без вычислений, **производные** – из базовых с помощью вычислений, **эффективные** – рассчитываются с учетом затраченного времени поиска.

Базовые

1. глубина поиска D – макс. число вершин, построенное до целевой или тупиковой вершины в дереве вариантов. самая длинная ветка, не важно, целевая или нет. 2 глоб. переменные с кол-вом шагов, макс левел и каррент левел.
2. длина пути решения L – число вершин, лежащих на минимальном из найденных путей решения.
3. общее число порожденных вершин N ( ++ при вызове порождающей процедуры)

Производные характеристики.

1. Разветвленность поиска R = N / L (чем меньше, тем лучше)
2. Направленность поиска. W = N1/L
3. Максмальное хранимое поддерево поиска M. Максим. число вершин, одновременно присутствовавших в оперативной памяти. (переменная для вычитания освобождаемой памяти).
4. Эффективность просмотра (анализа вершин). tc = T/N T- работы программы – как быстро обрабатывалась одна вершина.
5. Емкость поиска – обем ОП, занимаемый моделью данных V = Vb \* M, Vb – объем оперативной памяти, занимаемый моделью данных в байт.

Эффективные – с 1 по 5. отнесенные ко времени работы программы (разделить на время).

При оценке реальных алгоритмов используют выборку разнообразных задач. При этом варьируют параметры:

1. среднее число ходов, применимых к текущей ситуации (макс у меня 4),
2. емкость модели данных предметной области.
3. сложность задачи – длина пути решения, насколько отодвинуть робота от клада
4. эффективность порождения. даем писать задачи разным программистам.

Если оценивается какой-либо новый алгоритм и при этом получаются противоречивые характеристики, в этом случае выделяют основные признаки тех задач, для которых алгоритм работает эффективнее известных.

Метод альфа бета отсечений.

Метод заключается в отказе от просмотра вариантов ходов, на оторые противник может ответить с заведомо более сильным ходом, чем на ранее оцененные ходы. Альфа отсечение используется обоими соперниками при просмотре ходов макс. игрока в том случае, если какой-либо из рассматриваемых вариантов приписывает порождающей вершине оценку, заведомо меньшую, чем оценка предыдущей порождающей вершины.

Бета отсечение используется обоими соперниками при просмотре ходов мин. игрока в том случае, если какой-либо из рассматриваемых вариантов приписывает порождающей вершине оценку, заведомо большую, чем оценка предыдущих порождаемых вершин.

макс – альфа

мин – бета

# Заключение

* *некоторые результаты, (1-2 стр)*
* *а как можно было бы развить программу дальше*
* *а как бы вы сделали сейчас свою программу еще раз (как можно было по-другому)*

# Список использованной литературы

# Приложение.

* *Листинг основных модулей программы*

# 6. Оригинальное проектное решение. (научная новизна, какие применены решения)