УНИВЕРСАЛЬНОЕ ПОСОБИЕ ПО АЛГОРИТМАМ И СТРУКТУРАМ
ПОЛНЫЙ ИСХОДНЫЙ КОД ПРОЕКТА (ВСЕ ФАЙЛЫ, ВСЕ

Дата генерации:	04.09.2026, 21:52:24
Файлов исходного кода:	74
Суммарный объём кода:	954.5 KB
Глав/билетов в пособии:	30

ОГЛАВЛЕНИЕ: ГЛАВЫ И БИЛЕТЫ

1. 1. Дерево отрезков с операциями на отрезках [i
2. 2. Двумерная разреженная матрица (Sparse Table)
3. 3. Декартово дерево (Treap) [id: treap]
4. 4. Splay-дерево [id: splay-tree]
5. 5. Двумерная матрица префиксных сумм [id: pref
6. Планарность: K5, K3,3 и формула Эйлера [id: pl
7. 6. Графы. Представление, DFS, BFS [id: graph-d
8. 7. Графы. Планарные. Покраска [id: graph-plana
9. 8. Графы. Компоненты связности [id: graph-comp
10. 9. Графы. Топологическая сортировка [id: top-s
11. 10. Графы. Компоненты сильной связности (SCC)
12. 11. Графы. Мосты [id: graph-bridges]
13. 12. Графы. Точки сочленения [id: graph-articul
14. 13. Графы. Эйлеров цикл [id: graph-euler]
15. Билет 14. Дейкстра [id: dijkstra]
16. Билет 15. Форд-Беллман [id: bellman-ford]
17. Билет 16. Флойд-Уоршелл [id: floyd]
18. 17. Графы. Ускорения поиска [id: pathfinding-a
19. 17. Графы. Алгоритм Джонсона (Фокус с перевзвеши
20. 18. Графы. Остовное дерево. Краскал [id: mst-k
21. 19. Графы. Остовное дерево. Прима [id: mst-pri
22. 20. Графы. Остовное дерево. Борувка [id: mst-b
23. 21. Алгоритм Кнута-Морриса-Пратта (KMP) [id: s
24. 22. Z-функция [id: string-z-func]
25. Билет 23. Алгоритм Ахо-Корасик [id: aho-corasi

22. src/data/tickets/ticket1_5.ts
23. src/data/tickets/ticket14_20.ts
24. src/data/tickets/ticket21_24.ts
25. src/data/tickets/ticket6_13.ts

Интерактивные компоненты и симуляторы

26. src/components/ArticulationPointsViz.tsx
27. src/components/BellmanFordViz.tsx
28. src/components/BfsViz.tsx
29. src/components/ChapterImage.tsx
30. src/components/ChapterNav.tsx
31. src/components/ChapterView.tsx
32. src/components/DfsBridgesSimulator.tsx
33. src/components/DijkstraViz.tsx
34. src/components/DPVisualizer.tsx
35. src/components/EulerSimulator.tsx
36. src/components/ExpressQuiz.tsx
37. src/components/FloydViz.tsx
38. src/components/GraphTraversalViz.tsx
39. src/components/HeapViz.tsx
40. src/components/hints/demoKit.tsx
41. src/components/hints/demosGraph.tsx
42. src/components/hints/demosStruct.tsx
43. src/components/hints/MiniDemo.tsx
44. src/components/hints/SimulatorModal.tsx
45. src/components/hints/TermPopover.tsx
46. src/components/JohnsonViz.tsx
47. src/components/KosarajuViz.tsx
48. src/components/KruskalSimulator.tsx
49. src/components/MnemonicCards.tsx
50. src/components/MobileToc.tsx
51. src/components/Navbar.tsx
52. src/components/PlanarityDemo.tsx
53. src/components/QueueViz.tsx
54. src/components/SalmonAutomatonWidget.tsx
55. src/components/SegmentTreeVisualizer.tsx
56. src/components/Sidebar.tsx
57. src/components/Simulator.tsx

AI Agent Custom Instructions

You are the author of a Universal Educational Guide (Ун

CRITICAL DIRECTIVE:
Before you start creating new chapters, editing content
  iew_file` tool to read the skill instructions located
`.ai_skills/universal_guide_authoring/SKILL.md`

This file contains the mandatory rich HTML templates, s
  uired to maintain the high quality of this universal

Do NOT generate plain markdown chapters or invent new H

ФАЙЛ 2/74: GEMINI.md
КАТЕГОРИЯ: Конфигурация проекта | СТРОК: 16 | РАЗМЕР: 2

Gemini App Canvas Instructions

Ты — AI-ассистент, работающий в Gemini App Canvas (или
При запросе пользователя создать или сгенерировать нову
  строго придерживаться готовых шаблонов разметки,

Правила для малых страниц (Билетов / Глав)

1. **Единый стиль:** Всегда используй Tailwind CSS и HT
2. **Структура JSON/TS:** Каждая новая малая страница д
  l"`, `content: ...`) для вставки в `src/data/content.
3. **Строгая структура контента:**
   - Заголовок с цветным бейджем (например, `bg-indigo-
   - Определение/правило в центре (`border-blue-500` ил
   - Обязательная сетка аналогий (2 колонки: неформальн
   - "Душные" детали (код, формулы, сложные доказательс
4. **Не ломай верстку:** Не придумывай свои CSS-классы
  ерах `src/data/content.ts`. Не используй чистый markd

КАТЕГОРИЯ: Конфигурация проекта | СТРОК: 39 | РАЗМЕР: 1

```
{
  "name": "algv0-exam-guide",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "export:txt": "node scripts/export/collect-code.mjs
    "export:png": "node scripts/export/render-pages.mjs
    "export:pdf": "node scripts/export/build-pdf.mjs",
    "export": "npm run export:txt && npm run export:png
  },
  "dependencies": {
    "clsx": "2.1.1",
    "lucide-react": "^1.21.0",
    "motion": "^12.40.0",
    "pdfkit": "^0.20.1",
    "react": "19.2.6",
    "react-dom": "19.2.6",
    "tailwind-merge": "3.4.0"
  },
  "devDependencies": {
    "@tailwindcss/vite": "4.1.17",
    "@types/node": "22.19.17",
    "@types/pdfkit": "^0.17.6",
    "@types/react": "19.2.7",
    "@types/react-dom": "19.2.3",
    "@vitejs/plugin-react": "5.1.1",
    "esbuild": "^0.28.2",
    "tailwindcss": "4.1.17",
    "typescript": "5.9.3",
    "vite": "7.3.2",
    "vite-plugin-singlefile": "2.3.0"
```

Универсальное пособие • полный исходный код

```

Настройки через переменные окружения:

* `EXPORT_DENSITY=200` — DPI растеризации (по умолчанию
* `EXPORT_GRAYSCALE=1` — оттенки серого вместо 1-битного

### Автоматизация (CI)

`.github/workflows/rebuild-pdf.yml` запускается **на ка
пересобирает TXT → PNG → PDF, кладёт PNG-страницы и док
и коммитит обновлённые `public/export/full_code_all_pag
(с меткой `[skip ci]`, чтобы не зациклиться).

Приложение отдаёт эти файлы по ссылкам `/export/full_co

## Аудит контента

```bash
node scripts/audit-coverage.mjs        # какие темы без
node scripts/audit-coverage.mjs --md   # markdown-табли
```

Актуальный отчёт: [`TOPICS_AUDIT.md`](./TOPICS_AUDIT.md

Как устроен интерфейс

* **Содержание** — на десктопе боковая панель с поиском
  на телефоне — липкая строка «Содержание · N из M» и в
* **Переходы между темами** — кнопки «Назад / Вперёд» в
  «Предыдущая / Следующая тема» под текстом; на клавиат
* **Всплывающие подсказки по терминам** — текст главы а
  известные термины (BFS, куча, бор, суффиксная ссылка,
  подчёркиваются, а по наведению/тапу показывается карт
  мини-анимацией и кнопкой «Показать симулятор» (симуля
  Словарь — `src/data/glossary.ts`, мини-анимации — `sr
* **Реестр интерактивов** — `src/components/vizRegistry
  и для панели под главой, и для модалки из подсказки.
```

---

* Они помещены внутрь оберток с классом `<div class="im
* Все отрисовки находятся внутри векторного тега `<svg>
* Подпись к картинке, если она есть, идет сразу после S

Списки
* Стандартные списки оформлены через `<ul class="list-n

Валидный экспорт в Markdown (встроенный)
Экспорт перехватывает основные структурные компоненты H
1. Заголовки `<h2>` и `<h3>` заменяются на `##` и `###`
2. Элементы `.analogy-badge` получают выделение маркера
3. Теги `<svg>` заменяются строкой-заглушкой `[Иллюстр
   ла].
4. Элементы списка `<li>` получают маркер `-`.
5. Результат извлекается в виде текста и чистится от ли

ФАЙЛ 7/74: TOPICS_AUDIT.md
КАТЕГОРИЯ: Конфигурация проекта | СТРОК: 106 | РАЗМЕР:

Аудит покрытия тем: где нет «объяснения на пальцах» и

> Сгенерировано скриптом `node scripts/audit-coverage.m
> Критерии:
> * **Аналогия** — в HTML главы есть блок «Аналогия …»
> * **2D** — к главе привязан интерактивный визуализато
>
> Всего глав в пособии: **25**.

Уровень 1. Нет вообще ничего: ни аналогии, ни 2D

Глава	id	Что делать
Алгоритмы на карте	`alg-map`	Страница-заглушка (8

Универсальное пособие • полный исходный код

Уровень 4. Есть аналогия, но НЕТ 2D-визуализации (

> Закрыто: «5. Двумерная матрица префиксных сумм» пол

Глава	id	Чего не хватает
3. Декартово дерево (Treap)	`treap`	Нет ни визуал
8. Графы. Компоненты связности	`graph-components`
24. Классы сложности, сведение задач	`complexity-cl

Сводная таблица по всем главам

#	Глава	Аналогий	2D-виз.	SVG	Картинка
2	2. Двумерная разреженная матрица (Sparse Table)
3	3. Декартово дерево (Treap)	2	—	—	—
4	4. Splay-дерево	2		—	—
5	5. Двумерная матрица префиксных сумм	1		—
—	Планарность: K5, K3,3 и формула Эйлера	—	
6	6. Графы. Представление, DFS, BFS	4		—	—
8	8. Графы. Компоненты связности	1	—	—	—
9	9. Графы. Топологическая сортировка	1		—
10	10. Графы. Компоненты сильной связности (SCC)
12	12. Графы. Точки сочленения	1		—	—
14	14. Графы. Кратчайший путь. Дейкстра	3		—
15	15. Графы. Кратчайший путь. Форд–Беллман	1
16	16. Графы. Кратчайший путь. Флойд	—		—
17	17. Графы. Алгоритм Джонсона	—		—	—
18	18. Графы. Остовное дерево. Краскал	1		—
19	19. Графы. Остовное дерево. Прима	1		—
20	20. Графы. Остовное дерево. Борувка	1		—
21	21. Префикс-функция (π), КМП	4		—	—
22	22. Z-функция	4		—	—

```
----------------------------------------------------------------
    "noEmit": true,
    "jsx": "react-jsx",

    /* Path mapping */
    "baseUrl": ".",
    "paths": {
      "@/*": ["src/*"]
    },

    /* Linting */
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true
  },
  "include": ["src", "vite.config.ts"]
}


----------------------------------------------------------------
ФАЙЛ 9/74: vite.config.ts
КАТЕГОРИЯ: Конфигурация проекта | СТРОК: 33 | РАЗМЕР: 8
----------------------------------------------------------------

import path from "path";
import { fileURLToPath } from "url";
import tailwindcss from "@tailwindcss/vite";
import react from "@vitejs/plugin-react";
import { defineConfig } from "vite";
import { viteSingleFile } from "vite-plugin-singlefile"

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

// https://vite.dev/config/
export default defineConfig({
  plugins: [react(), tailwindcss(), viteSingleFile()],
  resolve: {
```

```
export default function App() {
  const [activeTab, setActiveTab] = useState<"guide" |
  const [activeChapterId, setActiveChapterId] = useStat
  const [modalVizId, setModalVizId] = useState<string |

  const index = Math.max(
    0,
    chapters.findIndex((c) => c.id === activeChapterId)
  );
  const activeChapter = chapters[index] ?? chapters[0];
  const prev = index > 0 ? chapters[index - 1] : undefi
  const next = index < chapters.length - 1 ? chapters[i

  const goTo = useCallback((id: string) => {
    setActiveChapterId(id);
    window.scrollTo({ top: 0, behavior: "smooth" });
  }, []);

  // Стрелки ← → листают темы (как на обучающих сайтах)
  useEffect(() => {
    const onKey = (e: KeyboardEvent) => {
      const target = e.target as HTMLElement | null;
      if (target && /^(INPUT|TEXTAREA|SELECT)$/.test(ta
      if (e.metaKey || e.ctrlKey || e.altKey) return;
      if (e.key === "ArrowRight" && next) goTo(next.id)
      if (e.key === "ArrowLeft" && prev) goTo(prev.id);
    };
    window.addEventListener("keydown", onKey);
    return () => window.removeEventListener("keydown",
  }, [goTo, next, prev]);

  const progress = useMemo(() => ((index + 1) / chapter

  return (
    <div className="min-h-screen bg-slate-950 text-slat
      <Navbar activeTab={activeTab} setActiveTab={setAc
```

```
                key={activeChapter.id}
                chapter={activeChapter}
                prev={prev}
                next={next}
                index={index}
                total={chapters.length}
                onGo={goTo}
                onOpenSimulator={setModalVizId}
              />
            </main>
          </div>
        </>
      ) : (
        <main className="p-4 md:p-8 lg:p-12 max-w-5xl m
          <Simulator />
        </main>
      )}

      <SimulatorModal vizId={modalVizId} onClose={() =>

      <footer className="p-8 text-center text-slate-600
        © 2026 Universal Educational Guide. Интерактивн
      </footer>
    </div>
  );
}
```

ФАЙЛ 11/74: src/hooks/useTermHints.ts
КАТЕГОРИЯ: Ядро приложения | СТРОК: 96 | РАЗМЕР: 3.7 KB

```
import { useEffect } from "react";
import { GLOSSARY_LABELS } from "../data/glossary";

/** Сколько раз один и тот же термин подсвечивается в г
const MAX_PER_TERM = 2;
```

```
    targets.push(current as Text);
    current = walker.nextNode();
  }

  for (const textNode of targets) {
    const text = textNode.nodeValue ?? "";
    re.lastIndex = 0;
    if (!re.test(text)) continue;
    re.lastIndex = 0;

    const frag = document.createDocumentFragment();
    let last = 0;
    let m: RegExpExecArray | null;

    while ((m = re.exec(text)) !== null) {
      const id = labelToId.get(m[1].toLowerCase());
      if (!id) continue;
      const used = counts.get(id) ?? 0;
      if (used >= MAX_PER_TERM) continue;
      counts.set(id, used + 1);

      if (m.index > last) frag.appendChild(document.crea
      const span = document.createElement("span");
      span.className = "term-hint";
      span.dataset.termId = id;
      span.tabIndex = 0;
      span.setAttribute("role", "button");
      span.setAttribute("aria-label", `Подсказка: ${m[1
      span.textContent = m[1];
      frag.appendChild(span);
      last = m.index + m[1].length;
    }

    if (last === 0) continue;
    if (last < text.length) frag.appendChild(document.c
    textNode.parentNode?.replaceChild(frag, textNode);
  }
}
```

```
  }
  .no-scrollbar {
    scrollbar-width: none;
  }
}

@keyframes pulse-gold {
  0%,
  100% {
    stroke: #eab308;
    stroke-width: 5;
    filter: drop-shadow(0 0 2px rgba(234, 179, 8, 0.5))
  }
  50% {
    stroke: #fef08a;
    stroke-width: 8;
    filter: drop-shadow(0 0 12px rgba(234, 179, 8, 0.9)
  }
}

.animate-pulse-gold {
  animation: pulse-gold 1.5s infinite;
}

@keyframes tocIn {
  from {
    transform: translateX(-100%);
  }
  to {
    transform: translateX(0);
  }
}

@keyframes hintIn {
  from {
    opacity: 0;
    transform: translateY(4px);
  }
```

```
}

.chapter-body :where(section, div, p, li) {
  min-width: 0;
}

.chapter-body :where(pre, code) {
  white-space: pre-wrap;
  word-break: break-word;
}

.chapter-body pre {
  overflow-x: auto;
  max-width: 100%;
  font-size: clamp(11px, 2.9vw, 13px);
}

.chapter-body table {
  display: block;
  width: 100%;
  overflow-x: auto;
  border-collapse: collapse;
  font-size: clamp(11px, 3vw, 14px);
}

.chapter-body summary {
  cursor: pointer;
  padding: 0.35rem 0;
}

@media (max-width: 767px) {
  .chapter-body :where(.grid) {
    grid-template-columns: minmax(0, 1fr) !important;
  }
  .chapter-body :where(h2) {
    font-size: 1.25rem !important;
    line-height: 1.3 !important;
  }
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import "./index.css";
import App from "./App";

createRoot(document.getElementById("root")!).render(
  <StrictMode>
    <App />
  </StrictMode>
);


--------------------------------------------------------------------------------
ФАЙЛ 14/74: src/types.ts
КАТЕГОРИЯ: Ядро приложения | СТРОК: 9 | РАЗМЕР: 153 B
--------------------------------------------------------------------------------

export interface Chapter {
  id: string;
  title: string;
  type: "html" | "markdown";
  content: string;
  description?: string;
  category?: string;
}


--------------------------------------------------------------------------------
ФАЙЛ 15/74: src/vite-env.d.ts
КАТЕГОРИЯ: Ядро приложения | СТРОК: 19 | РАЗМЕР: 324 B
--------------------------------------------------------------------------------

/// <reference types="vite/client" />

declare module '*.jpg' {
  const src: string;
```

```
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-blue-400
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-blue-300 italic
                <p class="text-xl font-mono text-white"
            ух отрезков: min(ST[L][k], ST[R - 2^k + 1][
            </div>
            <div class="grid grid-cols-1 xl:grid-cols-2
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4
            rder border-emerald-500 shadow-md">
                            Аналогия 1: Школьные линейки
                    </div>
                    <p class="text-slate-300 text-sm mb
            зок длины 7. Ты берешь две заготовленные за
            ь отрезок длины 7 (перекрывая друг друга по
            ничего складывать, просто пересекаем куски.
                </div>
                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4
             border-rose-500 shadow-md">
                            Аналогия 2: Ступени
                    </div>
                    <p class="text-slate-300 text-sm mb
            ужно покрыть отрезок, мы берем две самые бо
                </div>
            </div>
            <details class="bg-slate-900/40 rounded-lg
                <summary class="p-4 font-bold text-slat
            -b border-slate-700/50">
                        Код (Скрыто)
                </summary>
                <div class="p-5 text-sm text-slate-300
                    <pre class="bg-slate-950 p-4 rounde
int kx = log2(R2 - R1 + 1);
```

```
----------------------------------------------------------------------
                </div>
                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            border-rose-500 shadow-md">
                          Аналогия 2: Слияние капель (M
                    </div>
                    <p class="text-slate-300 text-sm mb
           вого). Мы просто смотрим на корни: у кого Y
            как потомок.</p>
                </div>
           </div>
           <details class="bg-slate-900/40 rounded-lg 
                <summary class="p-4 font-bold text-slat
           -b border-slate-700/50">
                     Код (Скрыто)
                </summary>
                <div class="p-5 text-sm text-slate-300 
                    <pre class="bg-slate-950 p-4 rounde
pair&lt;Node*, Node*&gt; split(Node* t, int x) {
    if (!t) return {nullptr, nullptr};
    if (t->val &lt;= x) {
        auto [L, R] = split(t->right, x);
        t->right = L;
        return {t, R};
    } else {
        auto [L, R] = split(t->left, x);
        t->left = R;
        return {L, t};
    }
}</pre>
                </div>
           </details>
        </div>
    </div>
</section>`,
  },
  {
    id: "splay-tree",
```

```
            <details class="bg-slate-900/40 rounded-lg i
                <summary class="p-4 font-bold text-slate
            -b border-slate-700/50">
                      Код (Скрыто)
                </summary>
                <div class="p-5 text-sm text-slate-300 
                    <pre class="bg-slate-950 p-4 rounde
// Zig, Zag
function rotate(x) {
    let p = x.parent;
    if (p.left === x) { // Right rotate
        p.left = x.right;
        x.right = p;
    } else { // Left rotate
        p.right = x.left;
        x.left = p;
    }
}</pre>
                </div>
            </details>
        </div>
    </div>
</section>`,
  },
  {
    id: "prefix-sums-2d",
    title: "5. Двумерная матрица префиксных сумм",
    type: "html",
    description: "Сжатие суммы подматрицы за O(1)",
    category: "Алгоритмы матрицы",
    content: `
<section id="prefix-sums-2d" class="mb-12 scroll-mt-10"
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-indigo-600 text-white px-4 py-1
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
```

```
---------------------------------------------------------------
                <p class="text-xl font-mono text-white":
            ый граф можно раскрасить 4 цветами.</p>
            </div>
            <div class="grid grid-cols-1 gap-6">
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 i
            rder border-emerald-500 shadow-md">
                            Аналогия 1: Карта мира
                    </div>
                    <p class="text-slate-300 text-sm mb
            . Границы не пересекаются в одной точке по-,
            ы не сливались цветами, Всегда можно всего `
                </div>
            </div>
        </div>
    </div>
</section>`,
  },
  {
    id: "top-sort",
    title: "9. Графы. Топологическая сортировка",
    type: "html",
    description: "Разложение задач по порядку выполнени.
    category: "Графы",
    content: `
<section id="top-sort" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3":
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-blue-400 
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-blue-300 italic 
                <p class="text-xl font-mono text-white":
            перед.</p>
            </div>
```

```
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-emerald-300 ital
                <p class="text-xl font-mono text-white"
            нвертированному в порядке top-sort.</p>
            </div>
            <div class="grid grid-cols-1 gap-6">
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4
            rder border-emerald-500 shadow-md">
                            Аналогия 1: Улицы с одностор
                    </div>
                    <p class="text-slate-300 text-sm mb
            БЫХ направлениях по односторонним улицам. К
            у на карте.</p>
                </div>
            </div>
        </div>
    </div>
</section>`,
  },
  {
    id: "graph-bridges",
    title: "11. Графы. Мосты",
    type: "html",
    description: "Рёбра, удаление которых расхреначивае
    category: "Графы. Продвинутые",
    content: `
<section id="graph-bridges" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-rose-400
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-rose-300 italic
                <p class="text-xl font-mono text-white"
```

```
                    <div class="absolute -top-3 left-4 i
            rder border-emerald-500 shadow-md">
                          Аналогия 1: Центральный роутер
                    </div>
                    <p class="text-slate-300 text-sm mb
             правильный роутер - и два сегмента офиса бо
                </div>
            </div>
        </div>
    </div>
</section>`,
  },
  {
    id: "graph-euler",
    title: "13. Графы. Эйлеров цикл",
    type: "html",
    description: "Пройти по всем ребрам ровно 1 раз.",
    category: "Графы",
    content: `
<section id="graph-euler" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3">
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-indigo-400
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-indigo-300 itali
                <p class="text-xl font-mono text-white">
            </div>
            <div class="grid grid-cols-1 gap-6">
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 i
            rder border-emerald-500 shadow-md">
                          Аналогия 1: Рисование не отры
                    </div>
                    <p class="text-slate-300 text-sm mb
```

```
            </div>
        </div>
    </div>
</section>`,
  },
  {
    id: "mst-kruskal",
    title: "18. Графы. Остовное дерево. Краскал",
    type: "html",
    description: "",
    category: "Графы. Остовные",
    content: `
<section id="mst-kruskal" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-emerald-4
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-emerald-300 ital
                <p class="text-xl font-mono text-white"
             (проверяем через DSU) - берем в ответ.</p>
            </div>
            <div class="grid grid-cols-1 gap-6">
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                            Аналогия: Прокладка интернет
                    </div>
                    <p class="text-slate-300 text-sm mb
            с самых дешевых дорог в стране". Он строит 
             единую сеть.</p>
                </div>
            </div>
        </div>
    </div>
```

```
    id: "mst-boruvka",
    title: "20. Графы. Остовное дерево. Борувка",
    type: "html",
    description: "",
    category: "Графы. Остовные",
    content: `
<section id="mst-boruvka" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-emerald-4
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-emerald-300 ital
                <p class="text-xl font-mono text-white"
            ается с соседом.</p>
            </div>
            <div class="grid grid-cols-1 gap-6">
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                            Аналогия: Племена
                    </div>
                    <p class="text-slate-300 text-sm mb
            изкому племени для объединения. К вечеру пл
            ства шлют гонцов к ближайшему чужому царств
                </div>
            </div>
        </div>
    </div>
</section>`,
  },
  {
    id: "string-kmp",
    title: "21. Алгоритм Кнута-Морриса-Пратта (KMP)",
    type: "html",
```

```
    int j = pi[i-1];
    while (j > 0 && s[i] != s[j]) j = pi[j-1];
    if (s[i] == s[j]) j++;
    pi[i] = j;
}</pre>
                </div>
            </details>
        </div>
    </div>
</section>`,
  },
  {
    id: "string-z-func",
    title: "22. Z-функция",
    type: "html",
    description: "",
    category: "Строки",
    content: `
<section id="string-z-func" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-indigo-600 text-white px-4 py-1
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-blue-400
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-blue-300 italic
                <p class="text-xl font-mono text-white"
            щегося с i.</p>
            </div>
            <div class="grid grid-cols-1 gap-6">
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4
            rder border-emerald-500 shadow-md">
                            Аналогия 1: Копипаста
                    </div>
                    <p class="text-slate-300 text-sm mb
```

```
---------------------------------------------------------------
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-purple-300 itali
                <p class="text-xl font-mono text-white"
            Сведение (Reduction) A->B: Если я умею реша
            </div>
            <div class="grid grid-cols-1 xl:grid-cols-2
                <!-- Аналогия 1 -->
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                            Аналогия 1: Судоку (P vs NP)
                    </div>
                    <p class="text-slate-300 text-sm mb
            ример сортировка чисел). Класс <b>NP</b> — 
            енную сетку (ответ), он БЫСТРО (за класс P)
                </div>
                <!-- Аналогия 2 -->
                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-purple-500 shadow-md">
                            Аналогия 2: Машина-переводчик
                    </div>
                    <p class="text-slate-300 text-sm mb
            ь отличный переводчик на английский (Сведен
            аче B, то B — это <b>NP-Полная</b> (сосредо
                </div>
            </div>
        </div>
    </div>
</section>`,
  },
];


---------------------------------------------------------------
ФАЙЛ 17/74: src/data/content_temp.ts
КАТЕГОРИЯ: Контент и данные пособия | СТРОК: 385 | РАЗМ
---------------------------------------------------------------
```

```
                <strong>Билет 13 (Различие пути и цикла
                <br>• <span class="text-green-400 font-
            шел-вошёл-вышел).
                <br>• <span class="text-yellow-400 font
             финиш).
            </p>
        </div>

        <div class="grid grid-cols-1 xl:grid-cols-2 gap
            <div class="bg-slate-800 p-6 rounded-lg bor
                <div class="absolute -top-3 left-4 bg-s
            order-green-500 shadow-md">
                        Путь: Нормальная прогулка
                </div>
                <p class="text-slate-300 text-sm mb-4">
                    Ты вышел из дома <strong>(нечётная)
            талый пришёл в бар <strong>(вторая нечётная
                    Домой вернуться не смог, потому что
                </p>
            </div>

            <div class="bg-slate-900 p-6 rounded-lg bor
                <div class="absolute -top-3 left-4 bg-r
            der-rose-500 shadow-md bg-slate-950">
                        Цикл: Гамильтонов синдром (болезн
                </div>
                <p class="text-slate-300 text-sm mb-4">
                    <strong>Гамильтон — это болезнь.</s
            ng> ровно раз и вернуться домой.
                    Ты обходишь все бары (вершины) ровн
                    Главное — зайти и выйти, не заходя
                    Никто не знает, как решать быстро.
                    <br><span class="text-xs text-rose-
            лноту).</span>
                </p>
            </div>
        </div>
```

```
                <div class="text-left font-mono text-sm
                    <p><strong class="text-white">tin[v
                    <p><strong class="text-white">low[v
                    <div class="p-3 bg-slate-950 rounde
                        <span class="text-rose-400 font
                        <p class="text-xl font-bold tex
                        <p class="text-xs text-slate-40
                    </div>
                </div>
            </div>

            <p class="text-slate-300 text-sm">
                Если из сына u и всех его потомков <str
                Единственная ниточка. Порвётся — граф р
            </p>
        </div>

        <div class="grid grid-cols-1 xl:grid-cols-2 gap
            <div class="bg-slate-800 p-6 rounded-lg bor
                <div class="absolute -top-3 left-4 bg-s
             border-emerald-500 shadow-md">
                        Мост: Единственная веревка
                </div>
                <p class="text-slate-300 text-sm mb-4">
                    Представь, что ты альпинист. Ты спу
                    Из пещеры u нет других выходов — то
                    Если верёвка (ребро v-u) оборвётся
                    Если бы из пещеры u был чёрный ход
                </p>
            </div>

            <div class="bg-slate-900 p-6 rounded-lg bor
                <div class="absolute -top-3 left-4 bg-r
            der-rose-500 shadow-md">
                        Аналогия: Кровеносный сосуд
                </div>
                <p class="text-slate-300 text-sm mb-4">
                    Граф — это кровеносная система. Реб
```

```
            low[v] = min(low[v], tin[to]);
        } else {
            // ребро дерева DFS
            dfs(to, v);                 // рекурсивный вы
            low[v] = min(low[v], low[to]);

            if (low[to] > tin[v]) {
                // ЭТО МОСТ!
                // bridge_list.push_back({v, to});
            }
        }
    }
}

void find_bridges() {
    timer = 0;
    visited.assign(n, false);
    tin.assign(n, -1);
    low.assign(n, -1);

    for (int i = 0; i &lt; n; ++i) {
        if (!visited[i]) dfs(i);
    }
}</pre>
                </div>
                <p class="text-xs text-slate-400">Запус
            </div>
        </details>
    </div>
</section>`,
  },
  {
    id: "planarity-euler-formula",
    title: "Планарность: K5, K3,3 и формула Эйлера",
    type: "html",
    content: `<section id="planarity-euler-formula" cla
    <div class="flex items-center mb-6">
        <span class="bg-blue-600 text-white px-4 py-1 r
```

```
            <div class="bg-slate-900 p-6 rounded-lg bor
                <div class="absolute -top-3 left-4 bg-r
            der-rose-500 shadow-md">
                        K3,3: 3 дома и 3 колодца
                </div>
                <p class="text-slate-300 text-sm mb-4">
                    Три дома хотят соединиться с тремя
                    Нарисовать без пересечений невозмож
                    У него <code class="text-white">V=6
            xt-white">9 ≤ 12</code> — проходит, но он в
                    Почему? Потому что у двудольного гр
                    Отсюда более строгое ограничение: <
                    <code class="text-white">9 &gt; 8</
                </p>
            </div>
        </div>

        <details class="bg-slate-900/40 rounded-lg bord
            <summary class="p-4 font-bold text-slate-40
            order-slate-700/50">
                  Доказательство непланарности K5 (Скры
            </summary>
            <div class="p-5 text-sm text-slate-300 spac
                <p class="text-blue-400">Доказательство
                <p>
                    Пусть K5 планарен. V = 5, E = 10.
                    <br>F = E - V + 2 = 10 - 5 + 2 = 7
                    <br>Каждая грань ограничена минимум
                    <br>Сумма длин граней = 2E = 20.
                    <br>Отсюда: 2E ≥ 3F ⇒ 20 ≥ 21 — <sp
                    <br>Значит, K5 не может быть планар
                </p>
            </div>
        </details>
    </div>
</section>`,
  },
```

```
                    Аналогия: Главный перекресток
                </div>
                <p class="text-slate-300 text-sm mb-4">
                    Представь город, где два района сое
            её перекроют (удалят вершину) — районы буд
            g>хрупкость</strong> системы.
                </p>
            </div>

            <div class="bg-slate-900 p-6 rounded-lg bor
                <div class="absolute -top-3 left-4 bg-e
            er border-emerald-500 shadow-md">
                    Телеком: Центральный свитч
                </div>
                <p class="text-slate-300 text-sm mb-4">
                    У тебя несколько компьютеров в одно
            абелем (мостом). Сами свитчи — это точки со
                </p>
            </div>
        </div>

        <details class="bg-slate-900/40 rounded-lg bord
            <summary class="p-4 font-bold text-slate-40
            order-slate-700/50">
                Душный мод: Код и корень DFS (Скрыто)
            </summary>
            <div class="p-5 text-sm text-slate-300 spac
                <p class="text-blue-400 font-bold">Особ
                <p>
                    Для стартовой вершины обхода правил
            то выше неё прыгать некуда). Поэтому для не
            у него <strong>больше 1 независимого ребенк
                </p>
                <div class="bg-slate-950 p-4 rounded-lg
                    <pre class="text-xs font-mono text-
void dfs(int v, int p = -1) {
    visited[v] = true;
    tin[v] = low[v] = timer++;
```

```
    </div>
</section>`,
  },
];
```

ФАЙЛ 18/74: src/data/content.ts
КАТЕГОРИЯ: Контент и данные пособия | СТРОК: 51 | РАЗМЕ

```
import { Chapter } from "../types";
import { graphChapters } from "./graphContent";
import { additionalTickets } from "./additionalTickets"
import { tickets14to20 } from "./tickets/ticket14_20";
import { tickets21to24 } from "./tickets/ticket21_24";
import { tickets6to13 } from "./tickets/ticket6_13";
import { chapters as newChapters } from "./content_temp

// We want to combine all of them, but overwrite 7, 11,
const merged = [
  ...graphChapters,
  ...additionalTickets,
  ...tickets6to13,
  ...tickets14to20,
  ...tickets21to24
].filter(c => c && c.id);

// Remove old 7, 11, 12, 13: graph-planar-colors, graph
const toRemove = ["graph-planar-colors", "graph-bridges

const dedup = new Map<string, Chapter>();
merged.forEach(c => {
  if (!toRemove.includes(c.id)) {
    // We prefer the newer versions (like tickets14_20 
    dedup.set(c.id, c);
  }
});
```

```
---------------------------------------------------------------
 */
export interface GlossaryEntry {
  id: string;
  /** Варианты написания в тексте (регистр не важен). *
  labels: string[];
  title: string;
  /** Объяснение «на пальцах» — 1–2 фразы. */
  short: string;
  /** Строгая формулировка / что важно не забыть. */
  formal?: string;
  complexity?: string;
  /** Какая мини-анимация показывается в карточке. */
  demo?: DemoKind;
  /** id полного симулятора (см. vizRegistry) — кнопка
  simulator?: string;
}

export const GLOSSARY: GlossaryEntry[] = [
  {
    id: "bfs",
    labels: ["BFS", "обход в ширину", "поиск в ширину",
    title: "BFS — обход в ширину",
    short:
      "Волна от источника: сначала все соседи, потом со
    formal:
      "Очередь (FIFO). Кладём старт, достаём вершину —
        у рёбер.",
    complexity: "O(V + E)",
    demo: "bfs",
    simulator: "queue-bfs",
  },
  {
    id: "dfs",
    labels: ["DFS", "обход в глубину", "поиск в глубину
    title: "DFS — обход в глубину",
    short:
      "Прёшь вперёд до упора, упёрся — откатываешься на
    formal:
```

```
    title: "DSU — система непересекающихся множеств",
    short:
      "«Кто твой староста?» Каждый элемент знает своего
    formal:
      "find со сжатием путей + union по рангу/размеру. 
    complexity: "почти O(1) — обратная функция Аккермана
    demo: "dsu",
    simulator: "mst-kruskal",
  },
  {
    id: "trie",
    labels: ["бор", "бора", "боре", "бору", "бором", "t
    title: "Бор (trie, префиксное дерево)",
    short:
      "Дерево, где путь от корня по буквам и есть слово
    formal: "Из корня по символам; в вершине хранится ф
    complexity: "поиск слова — O(|слова|)",
    demo: "trie",
    simulator: "aho-corasick",
  },
  {
    id: "bfs-on-trie",
    labels: [
      "обход дерева",
      "обход бора",
      "обход в ширину по бору",
      "какой-то обход",
      "обходом дерева",
    ],
    title: "Обход бора в ширину (тот самый «какой-то об
    short:
      "В Ахо-Корасике бор обходят именно BFS: суффиксну
        о есть строго по уровням.",
    formal:
      "Кладём в очередь детей корня (их ссылка = корень
          кладём с в очередь.",
    complexity: "O(суммарной длины словаря × алфавит)",
    demo: "trie-bfs",
```

```
    id: "prefix-sum",
    labels: ["префиксные суммы", "префиксная сумма", "пр
    title: "Префиксные суммы",
    short:
      "Заранее посчитанные «сколько всего набежало от на
    formal: "P[0]=0, P[i]=P[i-1]+a[i-1]. Сумма a[l..r] =
    complexity: "препроцессинг O(n), запрос O(1)",
    demo: "prefix-sum",
    simulator: "prefix-sums-2d",
  },
  {
    id: "sparse-table",
    labels: ["разреженная таблица", "sparse table", "pa
    title: "Разреженная таблица (метод двоичных подъёмо
    short:
      "Заранее считаем ответ для всех отрезков длины 1,
        м.",
    formal: "ST[i][j] = op(ST[i][j-1], ST[i+2^(j-1)][j-
    complexity: "препроцессинг O(n log n), запрос O(1)"
    demo: "sparse-table",
    simulator: "sparse-table",
  },
  {
    id: "segment-tree",
    labels: ["дерево отрезков", "дереве отрезков", "дер
    title: "Дерево отрезков",
    short:
      "Матрёшка из отрезков: корень отвечает за весь ма
        товых кусков.",
    formal: "Строится за O(n), запрос и обновление — O(
    demo: "segment-tree",
    simulator: "segment-trees",
  },
  {
    id: "bridge",
    labels: ["мост", "моста", "мосты", "мостов", "мосто
    title: "Мост",
    short:
```

```
  {
    id: "np",
    labels: ["NP-полная", "NP-полнота", "NP-трудная", "
    title: "Класс NP и NP-полнота",
    short:
      "NP — «решение проверить легко, найти трудно» (су
        ые в NP.",
    formal: "Задача NP-полна, если она в NP и к ней пол
    demo: "np",
  },
  {
    id: "potential",
    labels: ["потенциал", "потенциалы", "потенциалов",
    title: "Потенциалы (перевзвешивание Джонсона)",
    short:
      "Трюк «поднять все дороги на одинаковую высоту»: 
    formal: "w'(u,v) = w(u,v) + h(u) − h(v), где h — ра
      к путей сохраняется.",
    demo: "relax",
    simulator: "johnson-algo",
  },
  {
    id: "scc",
    labels: ["компонента сильной связности", "компонент
    title: "Компоненты сильной связности",
    short:
      "Группы вершин, где из каждой можно дойти до кажд
    formal: "Косарайю: DFS с сохранением порядка выхода
    complexity: "O(V + E)",
    demo: "scc",
    simulator: "scc-kosaraju",
  },
  {
    id: "prefix-function",
    labels: ["префикс-функция", "префикс-функции", "пре
    title: "Префикс-функция π (КМП)",
    short:
      "Для каждой позиции запоминаем: какой кусок начал
```

```
---------------------------------------------------------------------
      "Самый дешёвый маршрут из A в B. «Дешёвый» — по с
    formal:
      "Дейкстра — неотрицательные веса, O(E log V). Фор
        ы, O(V³).",
    demo: "relax",
    simulator: "dijkstra",
  },
  {
    id: "negative-cycle",
    labels: ["отрицательный цикл", "отрицательного цикл
    title: "Отрицательный цикл",
    short:
      "Круг, пройдя по которому вы становитесь «богаче»
    formal:
      "Форд–Беллман: если на V-й итерации ещё удаётся с
    demo: "relax",
    simulator: "bellman-ford",
  },
  {
    id: "greedy",
    labels: ["жадный алгоритм", "жадный", "жадно", "жад
    title: "Жадный алгоритм",
    short:
      "На каждом шаге хватаем то, что лучше прямо сейча
    formal:
      "Корректность обычно доказывается «леммой о разре
    demo: "mst",
    simulator: "mst-kruskal",
  },
  {
    id: "euler",
    labels: ["эйлеров цикл", "эйлеров путь", "эйлерова
    title: "Эйлеров путь и цикл",
    short:
      "Пройти по каждому ребру ровно один раз, не отрыв
    formal:
      "Связный граф: эйлеров цикл  все степени чётные;
    simulator: "euler-path-vs-cycle",
```

```
------------------------------------------------------------

/** Все метки, отсортированные от длинных к коротким (ч
export const GLOSSARY_LABELS: { label: string; id: stri
  e.labels.map((label) => ({ label, id: e.id }))
).sort((a, b) => b.label.length - a.label.length);


------------------------------------------------------------
ФАЙЛ 20/74: src/data/graphContent.ts
КАТЕГОРИЯ: Контент и данные пособия | СТРОК: 275 | РАЗМ
------------------------------------------------------------

import { Chapter } from "../types";

export const graphChapters: Chapter[] = [
  {
    id: "intro",
    title: "Обзор: Кратчайшие пути и Графы",
    type: "html",
    category: "Графы",
    content: `<section id="intro-content" class="mb-12 
    <div class="flex items-center mb-6">
        <span class="bg-indigo-600 text-white px-4 py-1
        <h2 class="text-3xl font-bold text-white">Обзор
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-blue-400 
            <p class="text-slate-300 text-sm mb-4">Пред
            оезда). Алгоритмы поиска кратчайшего пути п
            <div class="mt-8 mb-4">
               <div id="slot-mnemonic-cards"></div>
            </div>
        </div>
    </div>
</section>`
  },
  {
```

```
            охождение улицы), Дейкстра сломается, так ка
                </div>
            </div>

            <details class="bg-slate-900/40 rounded-lg b
                <summary class="p-4 font-bold text-slate
            -b border-slate-700/50">
                      Строгое доказательство (Скрыто)
                </summary>
                <div class="p-5 text-sm text-slate-300 
                    <p>Алгоритм использует приоритетную
            g V). Гарантирует корректность только для не
                </div>
            </details>

            <div id="slot-dijkstra-viz" class="mt-8"></
        </div>
    </div>
</section>`
  },
  {
    id: "bellman-ford",
    title: "Билет 15. Форд-Беллман",
    type: "html",
    category: "Графы",
    content: `<section id="bellman-ford-content" class=
    <div class="flex items-center mb-6">
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-3xl font-bold text-white">Форд-
    </div>

    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-rose-400 

            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-rose-300 italic 
                <p class="text-xl font-mono text-white"
```

```
        <h2 class="text-3xl font-bold text-white">Флойд
    </div>

    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-indigo-40

            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-indigo-300 itali
                <p class="text-xl font-mono text-white"
            </div>

            <div class="grid grid-cols-1 xl:grid-cols-2
                <!-- Аналогия 1 -->
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                            Аналогия 1: Автомагистрали
                    </div>
                    <p class="text-slate-300 text-sm mb
            евле долететь из Москвы в Париж, если мы сд
                    <div id="slot-chapter-image-floyd" 
                </div>

                <!-- Аналогия 2 -->
                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
             border-rose-500 shadow-md">
                            Время работы
                    </div>
                    <p class="text-slate-300 text-sm mb
             три вложенных цикла. Сложность O(V^3). Есл
                </div>
            </div>

            <div id="slot-floyd-viz" class="mt-8"></div
        </div>
    </div>
```

```
      <div class="bg-slate-900 p-6 rounded-lg border-2 b
        <div class="absolute -top-3 left-4 bg-rose-900 t
          -500 shadow-md">
            Аналогия 2: Матёшка (Терминальные)
        </div>
        <p class="text-slate-300 text-sm mb-4">Представ
          нашли и "he", потому что оно спрятано внутри т
          минальные ссылки</b> (term_link) — прямые мос
          p>
      </div>
    </div>

    <details class="bg-slate-900/40 rounded-lg border b
      <summary class="p-4 font-bold text-slate-400 outl
        pen:border-slate-700/50">
          Внутренняя структура автомата и код BFS (Скры
      </summary>
      <div class="p-5 text-sm text-slate-300 space-y-4"
        <p>Каждая вершина имеет таблицу переходов <code
           ссылку <code>term_link</code> (ближайшая мен
        <div class="bg-black/50 p-4 rounded-lg font-mon
struct Node {
    map&lt;char, int&gt; go;      // Предвычисленные пер
    int link = 0;           // Суффиксная ссылка: куда в
    int term_link = 0;      // Терминальная ссылка: мат
    bool is_terminal = false;
    vector&lt;string&gt; words;
};

// Расчет ссылок идет через BFS (level-order traversal)
// так как для вычисления link вершины v на глубине d,
// её предки на глубине d-1 уже должны иметь вычисленны
void buildLinks() {
    queue&lt;int&gt; q;
    // Корни и их дети
    for (auto const& [ch, child] : nodes[0].trie) {
        nodes[child].link = 0;
        q.push(child);
```

```
---------------------------------------------------------------
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-purple-40
            <p class="text-slate-300 text-sm mb-4">Когда
            ут быть большими дробными числами (долгота
            ву от 0 до N, и уже по ним строим графы.</p
        </div>
    </div>
</section>`
  }
];



---------------------------------------------------------------
ФАЙЛ 21/74: src/data/quizzes.ts
КАТЕГОРИЯ: Контент и данные пособия | СТРОК: 106 | РАЗМ
---------------------------------------------------------------

export interface QuizQuestion {
  question: string;
  options: string[];
  correctIndex: number;
  explanation: string;
}

export const quizzes: Record<string, QuizQuestion[]> =
  "euler-path-vs-cycle": [
    {
      question: "В связном графе ровно 2 вершины с нечё
      options: [
        "Есть эйлеров цикл (вернусь домой!)",
        "Есть эйлеров путь (из одной нечётной в другую)
        "Ни пути, ни цикла нет — это полный провал"
      ],
      correctIndex: 1,
      explanation: "Если нечётных вершин ровно 2 — это
        ётной, заканчиваем в другой."
    },
    {
```

```
        "Чтобы хранить глубину рекурсии"
      ],
      correctIndex: 1,
      explanation: "low[v] хранит минимальное время вхо
        поиска мостов и точек сочленения."
    },
    {
      question: "Какова временная сложность алгоритма п
      options: [
        "O(V²) — ищем мосты в квадрате",
        "O(V + E) — каждый элемент один раз",
        "O(2^V) — полный перебор"
      ],
      correctIndex: 1,
      explanation: "DFS обходит каждую вершину и каждое
    }
  ],
  "planarity-euler-formula": [
    {
      question: "Планарный граф имеет V=6, E=12. Возмож
      options: [
        "Да, если это двудольный граф",
        "Нет, нарушается неравенство E ≤ 3V - 6",
        "Да, если нет треугольных граней"
      ],
      correctIndex: 1,
      explanation: "E ≤ 3V - 6 => 12 ≤ 12 — на грани пр
         Так что такой граф непланарен."
    },
    {
      question: "Сколько граней у планарного графа с V=
      options: [
        "3 грани",
        "5 граней",
        "10 граней"
      ],
      correctIndex: 1,
      explanation: "F = E - V + 2 = 10 - 7 + 2 = 5. Не
```

```
    </div>

    <div class="space-y-8">
        <div id="intro" class="bg-slate-700/50 p-6 roun
            <h3 class="text-xl font-bold text-blue-400 

            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-blue-300 italic 
                <p class="text-xl font-mono text-white"
            omise (lazy).</p>
            </div>

            <div class="grid grid-cols-1 xl:grid-cols-2
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                            Аналогия 1: Корпоративная пре
                    </div>
                    <p class="text-slate-300 text-sm mb
            ы рассылать 10 000 писем, он пишет одно пис
             счетах сотрудников только тогда, когда сот
            женное обновление).</p>
                </div>

                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
             border-rose-500 shadow-md">
                            Аналогия 2: Матрешки-коробки
                    </div>
                    <p class="text-slate-300 text-sm mb
            ше, и так далее. Если нам нужно покрасить п
            Внутри всё синее!" на большую коробку. Опус
                </div>
            </div>

            <details class="bg-slate-900/40 rounded-lg 
                <summary class="p-4 font-bold text-slat
            -b border-slate-700/50">
```

```
            ух отрезков: min(ST[L][k], ST[R - 2^k + 1][
            </div>
            <div class="grid grid-cols-1 xl:grid-cols-2
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                          Аналогия 1: Школьные линейки
                    </div>
                    <p class="text-slate-300 text-sm mb
            зок длины 7. Ты берешь две заготовленные за
            ь отрезок длины 7 (перекрывая друг друга по
            ничего складывать, просто пересекаем куски.
                </div>
                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
             border-rose-500 shadow-md">
                          Аналогия 2: Ступени
                    </div>
                    <p class="text-slate-300 text-sm mb
            ужно покрыть отрезок, мы берем две самые бо
                </div>
            </div>
            <details class="bg-slate-900/40 rounded-lg 
                <summary class="p-4 font-bold text-slat
            -b border-slate-700/50">
                      Код (Скрыто)
                </summary>
                <div class="p-5 text-sm text-slate-300 
                    <pre class="bg-slate-950 p-4 rounde
int k = log2(R - L + 1);
int minimum = min(st[L][k], st[R - (1 << k) + 1][k]);</
                </div>
            </details>
        </div>
    </div>
</section>`
  },
  {
```

```
            <details class="bg-slate-900/40 rounded-lg i
                <summary class="p-4 font-bold text-slate
            -b border-slate-700/50">
                      Код (Скрыто)
                </summary>
                <div class="p-5 text-sm text-slate-300 
                    <pre class="bg-slate-950 p-4 rounde
pair&lt;Node*, Node*&gt; split(Node* t, int x) {
    if (!t) return {nullptr, nullptr};
    if (t->val &lt;= x) {
        auto [L, R] = split(t->right, x);
        t->right = L;
        return {t, R};
    } else {
        auto [L, R] = split(t->left, x);
        t->left = R;
        return {L, t};
    }
}</pre>
                </div>
            </details>
        </div>
    </div>
</section>`
  },
  {
    id: "splay-tree",
    title: "4. Splay-дерево",
    type: "html",
    description: "Дерево поиска, которое самобалансируе
    category: "Продвинутые структуры",
    content: `
<section id="splay-tree" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-indigo-600 text-white px-4 py-1
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
```

```
        p.left = x.right;
        x.right = p;
    } else { // Left rotate
        p.right = x.left;
        x.left = p;
    }
}</pre>
                </div>
            </details>

            <div class="bg-indigo-950/60 p-6 rounded-xl
                <h4 class="text-lg font-bold text-indig

                <div class="space-y-4 text-sm text-slat
                    <div>
                        <strong class="text-white block
            корень?</strong>
                        <p>Оно поднимет узел, на которо
            ска. За счет сдвига этого листа в корень, д
            /p>
                    </div>

                    <div>
                        <strong class="text-white block
                        <p>Если агент постоянно переклю
             находящимися на разных концах дерева, его
            ащая дерево в вытянутый бамбук для другого.
            . Постоянное свинг-переключение превратит п
                    </div>

                    <div>
                        <strong class="text-white block
                        <p>Хотя один поиск может стоить
            емах обладают прекрасным свойством: они не
             по пути. "Палочное" дерево прессуется в сб
            осов.</p>
                    </div>
                </div>
```

```
----------------------------------------------------------------
</section>`
  }
];


----------------------------------------------------------------
ФАЙЛ 23/74: src/data/tickets/ticket14_20.ts
КАТЕГОРИЯ: Билеты (учебный контент) | СТРОК: 248 | РАЗМ
----------------------------------------------------------------

import { Chapter } from '../../types';

export const tickets14to20: Chapter[] = [
  {
    id: "dijkstra",
    title: "14. Графы. Поиск кратчайшего пути. Дейкстра
    type: "html",
    category: "Графы. Пути",
    content: `
<section id="dijkstra-content" class="mb-12 scroll-mt-1
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-emerald-4
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-emerald-300 ital
                <p class="text-xl font-mono text-white"
            </div>
            <div class="grid grid-cols-1 xl:grid-cols-2
                <!-- Аналогия 1 -->
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                            Аналогия 1: Позитивное плани
                    </div>
```

```
                    <div class="absolute -top-3 left-4 b
            rder border-emerald-500 shadow-md">
                            ♂  Аналогия 1: Параноик
                    </div>
                    <p class="text-slate-300 text-sm mb
            еряет ВСЕ возможные дороги V-1 раз подряд. 
                </div>
                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 b
             border-rose-500 shadow-md">
                            Отрицательный цикл
                    </div>
                    <p class="text-slate-300 text-sm mb
            опали во временную петлю.</p>
                </div>
            </div>
            <div id="slot-bellman-viz" class="mt-8"></d
        </div>
    </div>
</section>`
  },
  {
    id: "floyd",
    title: "16. Графы. Поиск кратчайшего пути. Флойд",
    type: "html",
    category: "Графы. Пути",
    content: `
<section id="floyd-content" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-indigo-40
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-indigo-300 itali
                <p class="text-xl font-mono text-white"
```

```
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-amber-400
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-xl font-mono text-white"
            </div>
            <div class="grid grid-cols-1 gap-6 mb-6">
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            er border-amber-500 shadow-md">
                        ⚖  Суть перевзвешивания
                    </div>
                    <p class="text-slate-300 text-sm mb
                        Дейкстра (Билет 14) быстрая, но
            ает очень медленно. Мы хотим сделать <b>Джо
            е пути остались теми же.
                    </p>
                    <ol class="text-sm text-slate-300 s
                        <li>Добавляем <b>фиктивную верш
                        <li>Запускаем от неё <b>медленн
            b>«потенциалы»</b> <code class="text-pink-4
                        <li>Меняем старые веса: новое р
                        <li><b>Магия!</b> Все веса тепе
            ренние потенциалы сокращаются).</li>
                        <li>Теперь смело запускаем <b>б
                    </ol>
                </div>
            </div>
            <div id="slot-johnson-viz" class="mt-8"></d
        </div>
    </div>
</section>`
  },
  {
    id: "mst-kruskal",
    title: "18. Графы. Остовное дерево. Краскал",
    type: "html",
    category: "Графы. Остовные",
```

```
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-emerald-4
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-emerald-300 ital
                <p class="text-xl font-mono text-white"
             которое торчит из посещенных в непосещенны
            </div>
            <div class="grid grid-cols-1 gap-6">
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                        ☣  Аналогия: Заражение вирусом
                    </div>
                    <p class="text-slate-300 text-sm mb
            ли вирус). Мы стартуем из одного города, и
            Сеть растет только из одного связного куска
                </div>
            </div>
        </div>
    </div>
</section>`
  },
  {
    id: "mst-boruvka",
    title: "20. Графы. Остовное дерево. Борувка",
    type: "html",
    category: "Графы. Остовные",
    content: `
<section id="mst-boruvka" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
```

```
<section id="string-kmp" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-indigo-600 text-white px-4 py-1
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-blue-400
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-blue-300 italic
                <p class="text-xl font-mono text-white"
            сом, оканчивающимся в позиции i.</p>
            </div>

            <div class="grid grid-cols-1 xl:grid-cols-2
                <!-- Аналогия 1 -->
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4
            rder border-emerald-500 shadow-md">
                        ♂ Аналогия 1: Поиск без отка
                    </div>
                    <p class="text-slate-300 text-sm mb
                      Мы идём по строке слева направо.
            строки СЕЙЧАС закончился под моим пальцем?»
                      Представь, ты ищешь в тексте слов
            x</code>. Тупой алгоритм начнет искать зано
            </code>, а это начало нашего слова! Не надо
                    </p>
                </div>

                <!-- Аналогия 2 -->
                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4
             border-rose-500 shadow-md">
                        Аналогия 2: LLM стриминг
                    </div>
                    <p class="text-slate-300 text-sm mb
                      Для LLM, которая стримит токены
```

```
    int j = pi[i-1];
    while (j > 0 && s[i] != s[j]) j = pi[j-1];
    if (s[i] == s[j]) j++;
    pi[i] = j;
}</pre>
                </div>
            </details>
        </div>
    </div>
</section>`
  },
  {
    id: "string-z-func",
    title: "22. Z-функция — Сдвиг и наложение",
    type: "html",
    category: "Строки",
    content: `
<section id="string-z-func" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-indigo-600 text-white px-4 py-1
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-emerald-4
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-emerald-300 ital
                <p class="text-xl font-mono text-white"
            ca 0) и её суффиксом, начинающимся с i.</p>
            </div>

            <div class="grid grid-cols-1 xl:grid-cols-2
                <!-- Аналогия 1 -->
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                        Аналогия 1: Линейка-шаблон
                    </div>
```

```
                    </ul>
                </div>
            </details>
            <details class="bg-slate-900/40 rounded-lg i
                <summary class="p-4 font-bold text-slate
            -b border-slate-700/50">
                      Код C++ (Скрыто)
                </summary>
                <div class="p-5 text-sm text-slate-300 
                    <pre class="bg-slate-950 p-4 rounde
int l = 0, r = 0;
for (int i = 1; i &lt; n; i++) {
    if (i &lt;= r) z[i] = min(r - i + 1, z[i - l]);
    while (i + z[i] &lt; n && s[z[i]] == s[i + z[i]]) z
    if (i + z[i] - 1 > r) { l = i; r = i + z[i] - 1; }
}</pre>
                </div>
            </details>
        </div>
    </div>
</section>`
  },
  {
    id: "aho-corasick",
    title: "23. Алгоритм Ахо-Корасик",
    type: "html",
    category: "Строки",
    content: `
<section id="aho-corasick" class="mb-12 scroll-mt-10">
  <div class="flex items-center mb-6 flex-wrap gap-3">
    <span class="bg-indigo-600 text-white px-4 py-1 rou
    <h2 class="text-2xl sm:text-3xl font-bold text-whit
  </div>

  <div class="space-y-8">
    <div class="bg-slate-700/50 p-6 rounded-xl border-l
      <h3 class="text-xl font-bold text-blue-400 mb-4">
```

```
        if (v == 0 || t[v].p == 0) t[v].link = 0;
        else t[v].link = go(get_link(t[v].p), t[v].pch)
    }
    return t[v].link;
}

int go(int v, char c) {
    if (t[v].go[c] == -1) {
        if (t[v].next[c] != -1) t[v].go[c] = t[v].next[
        else t[v].go[c] = v == 0 ? 0 : go(get_link(v), 
    }
    return t[v].go[c];
}</pre>
        </div>
    </details>
  </div>
</section>`
  },
  {
    id: "complexity-classes",
    title: "24. Классы сложности, сведение задач",
    type: "html",
    category: "Теория сложности",
    content: `
<section id="complexity-classes" class="mb-12 scroll-mt
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-purple-600 text-white px-4 py-1
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-purple-40
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-purple-300 itali
                <p class="text-xl font-mono text-white"
            Сведение (Reduction) A->B: Если я умею реша
            </div>
            <div class="grid grid-cols-1 xl:grid-cols-2
```

```
----------------------------------------------------------------
    title: "6. Графы. Представление, DFS, BFS",
    type: "html",
    description: "Представление графов и базовые обходы
    category: "Графы. База",
    content: `
<section id="graph-dfs-bfs" class="mb-12 scroll-mt-10">
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        {/* База представлений */}
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-slate-300
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-slate-300 italic
                <p class="text-xl font-mono text-white"
            (V + E).</p>
            </div>
            <div class="grid grid-cols-1 xl:grid-cols-2
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                            Аналогия: Матрица = Таблица
                    </div>
                    <p class="text-slate-300 text-sm mb
             глупо, но проверка "знает ли Вася Петю" мг
                </div>
                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
             border-rose-500 shadow-md">
                            Аналогия: Списки = Контакты
                    </div>
                    <p class="text-slate-300 text-sm mb
            т. Оптимально для почти всех задач.</p>
                </div>
            </div>
        </div>
```

```
                        <li>Кратчайший путь в Невзвешен
                        <li>Алгоритм Ахо-Корасик (сборка
                        <li>Алгоритм Форда-Фалкерсона /
                        <li>Двунаправленный поиск для у
                    </ul>
                </div>
            </div>

            <details class="bg-slate-900/40 rounded-lg 
                <summary class="p-4 font-bold text-slate
            -b border-slate-700/50">
                      Код обходов C++ (Основа)
                </summary>
                <div class="grid grid-cols-1 md:grid-col
                    <pre class="bg-slate-950 p-3 rounde
// DFS (Рекурсия)
vector&lt;bool&gt; vis;
vector&lt;vector&lt;int&gt;&gt; adj;

void dfs(int v) {
    vis[v] = true;
    for (int u : adj[v]) {
        if (!vis[u]) dfs(u);
    }
}</pre>
                    <pre class="bg-slate-950 p-3 rounde
// BFS (Очередь)
queue&lt;int&gt; q;
q.push(start_node);
vis[start_node] = true;

while(!q.empty()) {
    int v = q.front(); q.pop();
    for (int u : adj[v]) {
        if (!vis[u]) {
            vis[u] = true;
            q.push(u);
        }
```

```
            </div>
        </div>
    </div>
</section>`,
  },
  {
    id: "graph-components",
    title: "8. Графы. Компоненты связности",
    type: "html",
    description: "Нахождение кусков графа",
    category: "Графы. База",
    content: `
<section id="graph-components" class="mb-12 scroll-mt-1
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-
    </div>
    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-emerald-4
            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-emerald-300 ital
                <p class="text-xl font-mono text-white"
            ество компонент.</p>
            </div>
            <div class="grid grid-cols-1 xl:grid-cols-2
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                            Аналогия 1: Острова
                    </div>
                    <p class="text-slate-300 text-sm mb
            карту — остались ли серые (неисследованные)
            ь через океан — столько и компонент.</p>
                </div>
            </div>
        </div>
    </div>
```

```
            получить опыт, нужна работа". Алгоритм выяв
                </div>
            </div>

            <details class="bg-slate-900/40 rounded-lg i
                <summary class="p-4 font-bold text-slate
            -b border-slate-700/50">
                      Душный мод: Код на C++ (Скрыто)
                </summary>
                <div class="p-5 text-sm text-slate-300 
                    <p class="text-indigo-400 font-bold
                    <div class="bg-slate-950 p-4 rounde
                        <pre class="text-xs font-mono t
vector&lt;int&gt; adj[MAXN];
bool visited[MAXN];
vector&lt;int&gt; ans;

void dfs(int v) {
    visited[v] = true;
    for (int to : adj[v]) {
        if (!visited[to]) {
            dfs(to);
        }
    }
    ans.push_back(v); // Добавляем в момент ВЫХОДА (pos
}

void topological_sort(int n) {
    for (int i = 0; i &lt; n; ++i) {
        if (!visited[i]) dfs(i);
    }
    reverse(ans.begin(), ans.end()); // Разворачиваем с
}</pre>
                    </div>
                </div>
            </details>
        </div>
    </div>
```

```
                <div class="bg-slate-800 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
            rder border-emerald-500 shadow-md">
                            Аналогия: Интриги в офисе
                    </div>
                    <p class="text-slate-300 text-sm mb
                        Сильная связность — это "клуб с
            ни в одной SCC. Добавить курьера Гошу, кото
            йти наружу (к директору), но обратно в клуб
                    </p>
                </div>

                <div class="bg-slate-900 p-6 rounded-lg
                    <div class="absolute -top-3 left-4 
             border-rose-500 shadow-md">
                           Связь с Билетом 12
                    </div>
                    <p class="text-slate-300 text-sm mb
                        <b>SCC (Билет 10)</b> склеивает
            </br><br>
                        <b>Точки сочленения (Билет 12)<
            ость</strong> монолита. Вырвешь точку сочле
                    </p>
                </div>
            </div>

            <details class="bg-slate-900/40 rounded-lg 
                <summary class="p-4 font-bold text-slat
            -b border-slate-700/50">
                      Душный мод: Код Алгоритм Косарайю
                </summary>
                <div class="p-5 text-sm text-slate-300 
                    <p class="text-emerald-400 font-bol
                    <ol class="list-decimal list-inside
                        <li>Запускаем DFS на исходном г
            li>
                        <li>Разворачиваем этот список (
                        <li>Переворачиваем все ребра в 
```

```
            </div>
        </div>
    </div>
</section>`,
  },
  {
    id: "graph-articulation",
    title: "12. Графы. Точки сочленения",
    type: "html",
    description: "Вершины, удаление которых убивает свя
    category: "Графы. Продвинутые",
    content: `
<section id="graph-articulation" class="mb-12 scroll-mt
    <div class="flex items-center mb-6 flex-wrap gap-3"
        <span class="bg-blue-600 text-white px-4 py-1 r
        <h2 class="text-2xl sm:text-3xl font-bold text-

    <div class="space-y-8">
        <div class="bg-slate-700/50 p-6 rounded-xl bord
            <h3 class="text-xl font-bold text-rose-400

            <div class="bg-slate-900 p-4 rounded-lg mb-
                <p class="text-lg text-rose-300 italic
                <div class="text-left font-mono text-sm
                    <p><strong class="text-white">Точка
            рой увеличивает число компонент связности.<
                    <div class="p-3 bg-slate-950 rounde
                        <span class="text-rose-400 font
                        <p class="text-xl font-bold tex
                        <p class="text-xs text-slate-40
                    </div>
                </div>
            </div>

            <p class="text-slate-300 text-sm">
                <strong>Важно:</strong> Это работает то
            епятся к точкам сочленения. То есть, <stron
```

```
            то выше неё прыгать некуда). Поэтому для не
            у него <strong>больше 1 независимого ребенк
                </p>
                <div class="bg-slate-950 p-4 rounded-lg
                    <pre class="text-xs font-mono text-
void dfs(int v, int p = -1) {
    visited[v] = true;
    tin[v] = low[v] = timer++;
    int children = 0;

    for (int to : adj[v]) {
        if (to == p) continue;
        if (visited[to]) {
            low[v] = min(low[v], tin[to]);
        } else {
            dfs(to, v);
            low[v] = min(low[v], low[to]);
            if (low[to] >= tin[v] && p != -1)
                IS_CUTPOINT(v);
            ++children;
        }
    }
    if (p == -1 && children > 1)
        IS_CUTPOINT(v);
}</pre>
                </div>
            </div>
        </details>

        <div class="bg-indigo-950/60 p-6 rounded-xl bor
            <h4 class="text-lg font-bold text-indigo-40
            <p class="text-slate-300 text-sm mb-4">
                Это глубокий дуальный мост между ориент
            </p>
            <ul class="list-disc list-inside text-sm te
                <li><strong class="text-rose-400">Точки
            ong> и показывают физическую уязвимость свя
                <li><strong class="text-emerald-400">SC
```

```
---------------------------------------------------------------
            а везде должно быть четное число (зашел-выш
                </div>
            </div>
        </div>
    </div>
</section>`,
  },
];
```

```
================================================================
РАЗДЕЛ: ИНТЕРАКТИВНЫЕ КОМПОНЕНТЫ И СИМУЛЯТОРЫ
================================================================
```

```
---------------------------------------------------------------
ФАЙЛ 26/74: src/components/ArticulationPointsViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
---------------------------------------------------------------

import React, { useState, useEffect } from "react";
import { Play, Pause, RotateCcw, Info } from "lucide-re

interface Node {
  id: number;
  x: number;
  y: number;
  label: string;
}

interface Edge {
  u: number;
  v: number;
}

const nodes: Node[] = [
  { id: 0, x: 250, y: 50, label: "0" },
  { id: 1, x: 100, y: 150, label: "1" },
  { id: 2, x: 250, y: 250, label: "2" },
```

```
  const [currentStepIndex, setCurrentStepIndex] = useSta
  const [isPlaying, setIsPlaying] = useState(false);

  useEffect(() => {
    const newSteps: StepInfo[] = [];
    let timer = 0;
    const visited = new Set<number>();
    const ap = new Set<number>();
    const currentTin = Array(nodes.length).fill(-1);
    const currentLow = Array(nodes.length).fill(-1);

    const recordStep = (desc: string, u: number | null)
      newSteps.push({
        desc,
        visited: new Set(visited),
        ap: new Set(ap),
        currentNode: u,
        tin: [...currentTin],
        low: [...currentLow],
      });
    };

    recordStep("Начало DFS (Тарьян) для поиска точек со

    const dfs = (v: number, p: number = -1) => {
      visited.add(v);
      currentTin[v] = currentLow[v] = timer++;
      let children = 0;

      recordStep(`Заходим в вершину ${v}. tin[${v}]=${c

      for (const to of adj[v]) {
        if (to === p) continue;

        if (visited.has(to)) {
          currentLow[v] = Math.min(currentLow[v], curre
          recordStep(
            `Обратное ребро ${v}-${to}. Обновляем low[$
```

```
        dfs(i);
      }
    }

    recordStep("Алгоритм завершен. Найдены все точки со
    setSteps(newSteps);
  }, []);

  const currentStep = steps[currentStepIndex] || {
    desc: "",
    visited: new Set(),
    ap: new Set(),
    currentNode: null,
    tin: [],
    low: [],
  };

  useEffect(() => {
    let interval: NodeJS.Timeout;
    if (isPlaying && currentStepIndex < steps.length -
      interval = setInterval(() => {
        setCurrentStepIndex((prev) => prev + 1);
      }, 1500);
    } else if (currentStepIndex >= steps.length - 1) {
      setIsPlaying(false);
    }
    return () => clearInterval(interval);
  }, [isPlaying, currentStepIndex, steps.length]);

  const togglePlay = () => setIsPlaying(!isPlaying);
  const reset = () => {
    setIsPlaying(false);
    setCurrentStepIndex(0);
  };

  return (
    <div className="bg-slate-900 rounded-xl border bord
      <div className="bg-slate-800 p-4 border-b border-
```

```
                (edge.u === 3 && edge.v === 4) ||
                (edge.u === 4 && edge.v === 3);

              return (
                <line
                  key={"edge-" + i}
                  x1={u.x}
                  y1={u.y}
                  x2={v.x}
                  y2={v.y}
                  stroke={isBridge ? "#f43f5e" : "#4755
                  strokeWidth={isBridge ? "4" : "2"}
                  strokeDasharray={isBridge ? "8,4" : "
                />
              );
            })}

            {/* Nodes */}
            {nodes.map((node) => {
              const isCurrent = currentStep.currentNode
              const isVisited = currentStep.visited.has
              const isAp = currentStep.ap.has(node.id);

              const fill = isCurrent
                ? "#3b82f6"
                : isAp
                  ? "#e11d48"
                  : isVisited
                    ? "#10b981"
                    : "#1e293b";
              const stroke = isCurrent
                ? "#60a5fa"
                : isAp
                  ? "#f43f5e"
                  : isVisited
                    ? "#34d399"
                    : "#334155";
              const r = isAp ? 24 : 20;
```

```
                    </text>
                    <text
                      x={node.x}
                      y={node.y + r + 20}
                      textAnchor="middle"
                      fill="#94a3b8"
                      fontSize="10px"
                    >
                      low:{" "}
                      {currentStep.low[node.id] >= 0
                        ? currentStep.low[node.id]
                        : "-"}
                    </text>
                  </>
                )}
              </g>
            );
          })}
        </svg>
      </div>

      <div className="bg-slate-950 p-6 border-t lg:bo
        <h4 className="text-sm font-bold text-slate-4
          Статус алгоритма
        </h4>

        <div className="bg-slate-900 border border-sl
          <p className="text-white font-medium min-h-
            {currentStep.desc}
          </p>
        </div>

        <div className="space-y-4 flex-1 overflow-y-a
          <div className="mb-4">
            <h5 className="text-xs text-slate-500 fon
            <div className="flex items-center gap-2 t
              <div className="w-3 h-3 rounded-full bg
              Точка сочленения
```

```
                  className="bg-rose-500 h-full transitio
                  style={{
                    width: `${(currentStepIndex / Math.ma
                  }}
                />
              </div>
            </div>
          </div>
        </div>
      );
    };


```

ФАЙЛ 27/74: src/components/BellmanFordViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО

```
import { useState, useEffect } from 'react';

interface Node { id: string; x: number; y: number; name
interface Edge { u: string; v: string; w: number; }
interface Step {
  iteration: number;
  checkingEdge: { u: string; v: string; w: number } | n
  distances: { [key: string]: number };
  previous: { [key: string]: string | null };
  log: string;
  hasNegativeCycle?: boolean;
  activeCodeLine: number;
}

export default function BellmanFordViz() {
  const [hasCycle, setHasCycle] = useState(false);
  const [steps, setSteps] = useState<Step[]>([]);
  const [currentStepIndex, setCurrentStepIndex] = useSt
  const [isPlaying, setIsPlaying] = useState(false);
```

```
  const codeLines = [
    { line: 1, text: 'function bellman_ford(graph, star
    { line: 2, text: '    dist = {v: ∞}; dist[start] = 
    { line: 3, text: '    for i from 1 to V - 1: // |V|
    { line: 4, text: '        for (u, v, weight) in E:'
    { line: 5, text: '            if dist[u] + weight <
    { line: 6, text: '                dist[v] = dist[u]
    { line: 7, text: '    for (u, v, weight) in E: // П
    { line: 8, text: '        if dist[u] + weight < dis
    { line: 9, text: '            return "ОБНАРУЖЕН ОТР
    { line: 10, text: '    return dist' },
  ];

  useEffect(() => {
    const simulationSteps: Step[] = [];
    let dist: { [key: string]: number } = { S: 0, A: 99
    let prev: { [key: string]: string | null } = { S: n

    simulationSteps.push({
      iteration: 0, checkingEdge: null,
      distances: { ...dist }, previous: { ...prev },
      log: '  Фура заведена. Начинаем с Базы (S). Расст
      activeCodeLine: 2,
    });

    const vCount = nodes.length;
    for (let iter = 1; iter < vCount; iter++) {
      let anyUpdate = false;
      for (const edge of edges) {
        const uDist = dist[edge.u];
        const vDist = dist[edge.v];

        let updated = false;
        if (uDist !== 999 && uDist + edge.w < vDist) {
          dist = { ...dist, [edge.v]: uDist + edge.w };
          prev = { ...prev, [edge.v]: edge.u };
          updated = true;
```

```
      if (!cycleFound) {
        simulationSteps.push({
          iteration: vCount, checkingEdge: null,
          distances: { ...dist }, previous: { ...prev }
          log: '  Проверка завершена. Ни одно ребро не
          activeCodeLine: 10,
        });
      }
    }

    setSteps(simulationSteps);
    setCurrentStepIndex(0);
    setIsPlaying(false);
  }, [hasCycle]);

  useEffect(() => {
    let timer: any = null;
    if (isPlaying && currentStepIndex < steps.length -
      timer = setTimeout(() => setCurrentStepIndex((p) =
    } else if (currentStepIndex >= steps.length - 1) {
      setIsPlaying(false);
    }
    return () => clearTimeout(timer);
  }, [isPlaying, currentStepIndex, steps.length]);

  const currentStep = steps[currentStepIndex] || null;
  if (!currentStep) return <div>Загрузка...</div>;

  return (
    <div className="bg-slate-800/90 p-6 rounded-2xl bor
      <div className="flex flex-wrap items-center justi
        <div className="flex items-center gap-3">
          <div className="p-2.5 bg-blue-600 text-white
            <span className="text-2xl"> </span>
          </div>
          <div>
            <h3 className="text-2xl font-bold text-whit
            <p className="text-sm text-blue-300">Полный
```

```
            <marker id="arrow-bf-normal" markerWidth=
          <path d="M0,0 L6,3 L0,6 z" fill="#64748b" />
            <marker id="arrow-bf-active" markerWidth=
          <path d="M0,0 L6,3 L0,6 z" fill="#f59e0b" />
            <marker id="arrow-bf-cycle" markerWidth="
          path d="M0,0 L6,3 L0,6 z" fill="#e11d48" />
          </defs>

          {edges.map((edge, idx) => {
            const uNode = nodes.find((n) => n.id === 
            const vNode = nodes.find((n) => n.id === 
            const isChecking = currentStep.checkingEd
          v;
            const isNeg = edge.w < 0;
            const isCyclePart = hasCycle && ((edge.u =
           && edge.v === 'D'));

            let strokeColor = '#475569';
            let marker = 'url(#arrow-bf-normal)';
            if (isChecking) { strokeColor = '#f59e0b'
            else if (currentStep.hasNegativeCycle && 

            return (
              <g key={idx}>
                <line x1={uNode.x} y1={uNode.y} x2={v
          asNegativeCycle && isCyclePart) ? 4 : 2} ma
          g ? '4,4' : 'none'} />
                <circle cx={(uNode.x + vNode.x) / 2} 
          Checking ? '#f59e0b' : isNeg ? '#f43f5e' : 
                <text x={(uNode.x + vNode.x) / 2} y={
          8fafc'} fontSize="11" fontWeight="bold" tex
              </g>
            );
          })}

          {nodes.map((node) => {
            const dist = currentStep.distances[node.i
            const isCheckingNode = currentStep.checki
```

```
                  }))}
                </div>
              ))}
            </div>
          </div>
        </div>

        <div className="flex flex-col lg:flex-row gap-6">
          {/* Таблица расстояний */}
          <div className="w-full lg:w-1/2 bg-rose-950/10
            <div>
              <h4 className="text-lg font-bold text-rose-
              <div className="grid grid-cols-4 gap-2 text
                 {nodes.map(n => {
                   const d = currentStep.distances[n.id];
                   const isTgt = currentStep.checkingEdge
                   return (
                     <div key={n.id} className={`p-2 roun
            : 'border-slate-700/60 bg-slate-800/40'}`}
                       <div className="text-xs text-slate
                       <div className={`font-mono font-bo
              v>
                     </div>
                   );
                 }))}
              </div>
            </div>
          </div>

          {/* Код алгоритма */}
          <div className="w-full lg:w-1/2 bg-slate-900/60
            <div>
              <h4 className="text-lg font-bold text-slate
              <div className="bg-slate-950/80 rounded-lg
                {codeLines.map(item => (
                  <div key={item.line} className={`py-1.5
              e === item.line ? 'bg-blue-900/80 text-ambe
                    <span className="text-slate-600 w-5 s
```

```
};

interface Step {
  activeLine: number;
  queue: number[];
  visited: number[];
  currentNode: number | null;
  logs: string;
}

const generateBfsSteps = (): Step[] => {
  const steps: Step[] = [];
  const queue: number[] = [];
  const visited = new Set<number>();

  // Init
  steps.push({
    activeLine: 1,
    queue: [],
    visited: [],
    currentNode: null,
    logs: "Инициализация: начинаем с корня (0)."
  });

  queue.push(0);
  visited.add(0);
  steps.push({
    activeLine: 2,
    queue: [...queue],
    visited: Array.from(visited),
    currentNode: null,
    logs: "Добавляем стартовую вершину 0 в очередь и по
  });

  while (queue.length > 0) {
    steps.push({
      activeLine: 4,
      queue: [...queue],
```

```
    } else {
        steps.push({
            activeLine: 6,
            queue: [...queue],
            visited: Array.from(visited),
            currentNode: curr,
            logs: `Соседей нет или все уже посещены.`
        });
    }
  }

  steps.push({
    activeLine: 12,
    queue: [],
    visited: Array.from(visited),
    currentNode: null,
    logs: "Очередь пуста. Алгоритм завершен!"
  });

  return steps;
};

const STEPS = generateBfsSteps();

export default function BfsViz() {
  const [stepIdx, setStepIdx] = useState(0);
  const [isPlaying, setIsPlaying] = useState(false);

  const step = STEPS[stepIdx];

  const handleNext = useCallback(() => {
    setStepIdx(prev => Math.min(prev + 1, STEPS.length
  }, []);

  // @ts-expect-error зарезервировано под кнопку «назад
  const handlePrev = useCallback(() => {
    setStepIdx(prev => Math.max(prev - 1, 0));
  }, []);
```

```
            className="p-2 text-emerald-400 hover:text-
            ed:hover:bg-transparent"
            title="Шаг вперед"
          >
            <StepForward className="w-5 h-5" />
          </button>
        </div>
      </div>

      <div className="grid grid-cols-1 lg:grid-cols-2 g
        {/* Визуал графа */}
        <div className="bg-slate-800 border border-slat
          50px]">
          <svg className="w-full h-full min-h-[350px]"
            {/* Рёбра */}
            {EDGES.map((e, idx) => {
              const n1 = NODES.find(n => n.id === e.fro
              const n2 = NODES.find(n => n.id === e.to)
              const isVisited = step.visited.includes(n
              return (
                <line
                  key={idx}
                  x1={n1.x} y1={n1.y} x2={n2.x} y2={n2.
                  className={`transition-all duration-5
            '}`}
                />
              );
            })}

            {/* Вершины */}
            {NODES.map(n => {
              const isCurrent = step.currentNode === n.
              const isInQueue = step.queue.includes(n.i
              const isVisited = step.visited.includes(n

              let fill = '#1e293b'; // slate-800
              let stroke = '#475569'; // slate-600
              let scale = 1;
```

```
            })}
          </svg>
        </div>

        {/* Код и Очередь */}
        <div className="flex flex-col gap-4">
          <div className="bg-slate-900 border border-sla
            er">
            <div className="bg-slate-800 text-slate-400
            se tracking-wider">
              <span>bfs(graph, start)</span>
              <span classNamе="text-blue-400">Шаг {step
            </div>
            <div classNamе="p-4">
              {[
                { line: 1, code: 'queue = Queue()' },
                { line: 2, code: 'queue.push(start); vi
                { line: 3, code: '' },
                { line: 4, code: 'while not queue.isEmp
                { line: 5, code: '  node = queue.pop()'
                { line: 6, code: '  for neighbor in gra
                { line: 7, code: '    if neighbor not i
                { line: 8, code: '        visited.add(ne
                { line: 9, code: '        queue.push(nei
              ].map((cl) => {
                const isActive = step.activeLine === cl
                return (
                  <div key={cl.line} classNamе={`px-2 p
            d border-l-2 border-blue-400 -ml-0.5' : 'te
                    <span classNamе="opacity-40 w-6 inl
                    <span classNamе="whitespace-pre">{c
                  </div>
                );
              })}
            </div>
          </div>

          <div classNamе="bg-slate-800/80 border border
```

```
  dijkstra: {
    src: dijkstraImg,
    alt: 'Добрый Дейкстра',
    caption: '  Добрый — только позитив!',
    borderColor: 'border-blue-500/30',
    captionColor: 'text-blue-400',
  },
  'bellman-ford': {
    src: bellmanImg,
    alt: 'Фура по болоту',
    caption: '  Тяжёлая, но ей пофиг на ямы!',
    borderColor: 'border-rose-500/30',
    captionColor: 'text-rose-400',
  },
  floyd: {
    src: floydImg,
    alt: 'Фсе ко Фсем Флойд',
    caption: '  Все связаны со всеми!',
    borderColor: 'border-emerald-500/30',
    captionColor: 'text-emerald-400',
  },
};

export default function ChapterImage({ vizType }: { viz
  const info = imageMap[vizType];
  if (!info) return null;

  return (
    <div>
      <div className={`rounded-2xl overflow-hidden bord
        <img src={info.src} alt={info.alt} className="w
      </div>
      <p className={`text-center text-xs mt-2 font-bold
    </div>
  );
}
```

```
        </span>
        <button
          disabled={!next}
          onClick={() => next && onGo(next.id)}
          title={next ? next.title : "Это последняя тем
          className="flex items-center gap-1 px-2.5 py-
            :bg-slate-700 enabled:hover:text-white disa
        >
          <span className="hidden sm:inline">Вперёд</sp
          <ChevronRight className="w-4 h-4" />
        </button>
      </div>
    );
  }

  return (
    <nav className="grid grid-cols-1 sm:grid-cols-2 gap
      {prev ? (
        <button
          onClick={() => onGo(prev.id)}
          className="group text-left p-4 rounded-xl bg-
            transition-all"
        >
          <div className="flex items-center gap-1.5 tex
            400 mb-1">
            <ChevronLeft className="w-3.5 h-3.5" /> Пре
          </div>
          <div className="text-sm font-bold text-slate-
        </button>
      ) : (
        <div className="hidden sm:block" />
      )}

      {next && (
        <button
          onClick={() => onGo(next.id)}
          className="group text-right p-4 rounded-xl bg
             transition-all sm:col-start-2"
```

```
export const ChapterView: React.FC<Props> = ({ chapter,
  const contentRef = useRef<HTMLDivElement>(null);
  const [anchor, setAnchor] = useState<HintAnchor | nul
  const hideTimer = useRef<number | null>(null);

  useTermHints(contentRef, [chapter.id]);

  const cancelHide = useCallback(() => {
    if (hideTimer.current) {
      window.clearTimeout(hideTimer.current);
      hideTimer.current = null;
    }
  }, []);

  const scheduleHide = useCallback(() => {
    cancelHide();
    hideTimer.current = window.setTimeout(() => setAnch
  }, [cancelHide]);

  useEffect(() => {
    setAnchor(null);
  }, [chapter.id]);

  const open = useCallback(
    (el: HTMLElement) => {
      const termId = el.dataset.termId;
      if (!termId) return;
      cancelHide();
      setAnchor({ termId, rect: el.getBoundingClientRec
    },
    [cancelHide]
  );

  const handlePointerOver = (e: React.MouseEvent) => {
    if (isTouch()) return;
    const el = (e.target as HTMLElement).closest<HTMLEl
    if (el) open(el);
  };
```

```
          <span>
            Подчёркнутые термины — интерактивные: наведи
            объяснение на пальцах, мини-анимацию и кноп
          </span>
        </p>

        {viz && (
          <section id="chapter-viz" className="mt-8 scr
            <div className="flex items-center justify-b
              <h3 className="flex items-center gap-2 te
                <Sparkles className="w-4 h-4 text-indig
                Демонстрация: {viz.title}
              </h3>
              <button
                onClick={() => onOpenSimulator(chapter.
                className="flex items-center gap-1.5 te
            ate-300 hover:text-white hover:border-indig
              >
                <Maximize2 className="w-3.5 h-3.5" /> P
              </button>
            </div>
            {viz.hint && <p className="text-xs text-sla
            <viz.Component />
          </section>
        )}

        <ChapterNav prev={prev} next={next} index={inde
      </article>

      <TermPopover
        anchor={anchor}
        onClose={() => setAnchor(null)}
        onOpenSimulator={(id) => {
          setAnchor(null);
          onOpenSimulator(id);
        }}
        onCardEnter={cancelHide}
        onCardLeave={scheduleHide}
```

```
  { line: "}", highlight: "normal" },
];

interface DfsStep {
  currentNode: number | null;
  visitedIndices: number[];
  tin: Record<number, number>;
  low: Record<number, number>;
  stack: number[];
  bridges: string[];
  log: string;
  activeEdge: [number, number] | null;
  backEdge: [number, number] | null;
  activeCodeLine: number[];
}

const GRAPH_NODES = [
  { id: 0, label: "A", x: 100, y: 120 },
  { id: 1, label: "B", x: 260, y: 120 },
  { id: 2, label: "C", x: 180, y: 200 },
  { id: 3, label: "D", x: 180, y: 300 },
  { id: 4, label: "E", x: 100, y: 370 },
  { id: 5, label: "F", x: 260, y: 370 },
  { id: 6, label: "G", x: 340, y: 200 }
];

const GRAPH_EDGES = [
  { u: 0, v: 1, key: "0-1" },
  { u: 1, v: 2, key: "1-2" },
  { u: 2, v: 0, key: "0-2" },
  { u: 2, v: 3, key: "2-3" },
  { u: 3, v: 4, key: "3-4" },
  { u: 4, v: 5, key: "4-5" },
  { u: 5, v: 3, key: "3-5" },
  { u: 1, v: 6, key: "1-6" }
];

const DFS_STEPS: DfsStep[] = [
```

```
    activeEdge: null, backEdge: [2,0],
    log: "Видим соседа A (уже посещён)! Обратное ребро
    activeCodeLine: [7,8,9]
  },
  {
    currentNode: 3, visitedIndices: [0,1,6,2,3],
    tin: {0:1,1:2,6:3,2:4,3:5}, low: {0:1,1:2,6:3,2:1,3
    activeEdge: [2,3], backEdge: null,
    log: "Идём C→D. tin[D]=5, low[D]=5.",
    activeCodeLine: [4,5,7,10,11]
  },
  {
    currentNode: 4, visitedIndices: [0,1,6,2,3,4],
    tin: {0:1,1:2,6:3,2:4,3:5,4:6}, low: {0:1,1:2,6:3,2
    activeEdge: [3,4], backEdge: null,
    log: "D→E. tin[E]=6, low[E]=6.",
    activeCodeLine: [4,5,7,10,11]
  },
  {
    currentNode: 5, visitedIndices: [0,1,6,2,3,4,5],
    tin: {0:1,1:2,6:3,2:4,3:5,4:6,5:7}, low: {0:1,1:2,6
    activeEdge: [4,5], backEdge: null,
    log: "E→F. tin[F]=7, low[F]=7.",
    activeCodeLine: [4,5,7,10,11]
  },
  {
    currentNode: 5, visitedIndices: [0,1,6,2,3,4,5],
    tin: {0:1,1:2,6:3,2:4,3:5,4:6,5:7}, low: {0:1,1:2,6
    activeEdge: null, backEdge: [5,3],
    log: "Из F видим D (посещён). Обратное ребро (F,D).
    activeCodeLine: [7,8,9]
  },
  {
    currentNode: 4, visitedIndices: [0,1,6,2,3,4,5],
    tin: {0:1,1:2,6:3,2:4,3:5,4:6,5:7}, low: {0:1,1:2,6
    activeEdge: null, backEdge: null,
    log: "Возврат F→E. low[E]=min(6,5)=5. (E,F) — не мо
    activeCodeLine: [11,13,16,17]
```

```
export function DfsBridgesSimulator() {
  const [dfsIdx, setDfsIdx] = useState(0);
  const [dfsAuto, setDfsAuto] = useState(false);
  const dfsTimer = useRef<NodeJS.Timeout | null>(null);

  useEffect(() => {
    if (dfsAuto) {
      dfsTimer.current = setInterval(() => {
        setDfsIdx(prev => {
          if (prev >= DFS_STEPS.length - 1) { setDfsAut
          return prev + 1;
        });
      }, 2500);
    }
    return () => { if (dfsTimer.current) clearInterval(
  }, [dfsAuto]);

  const step = DFS_STEPS[dfsIdx];

  return (
    <div className="space-y-4">
      <div className="flex items-center justify-between
        <h4 className="text-base font-bold text-white">
        <div className="flex items-center gap-1 bg-slat
          <button onClick={() => { setDfsAuto(false); s
            disabled={dfsIdx === 0}
            className="p-1.5 hover:bg-slate-700 rounded
            <Undo className="h-3.5 w-3.5" />
          </button>
          <button onClick={() => setDfsAuto(!dfsAuto)}
            className={`px-2.5 py-1 rounded text-[10px]
              dfsAuto ? "bg-rose-600 text-white" : "bg-
            }`}>
            {dfsAuto ? <><Pause className="h-3 w-3" />
          </button>
          <button onClick={() => { setDfsAuto(false); s
            disabled={dfsIdx === DFS_STEPS.length-1}
            className="p-1.5 hover:bg-slate-700 rounded
```

```
              if (!n) return null;
              return <circle cx={n.x} cy={n.y} r={18} f
            >;
            })()}
            {GRAPH_NODES.map(n => {
              const cur = step.currentNode === n.id;
              const vis = step.visitedIndices.includes(
              const st = step.stack.includes(n.id);
              return <g key={n.id}>
                <circle cx={n.x} cy={n.y} r={12}
                  fill={cur ? "#10b981" : st ? "#064e3b
                  stroke={cur ? "#fff" : st ? "#34d399"
                  strokeWidth={2.5}
                  className="transition-all duration-30
                <text x={n.x} y={n.y+3} textAnchor="mid
            label}</text>
                {vis && (
                  <>
                    <rect x={n.x-18} y={n.y-28} width={
            "transition-all duration-300" />
                    <text x={n.x} y={n.y-20} textAnchor=
            } l:{step.low[n.id]||"?"}</text>
                  </>
                )}
              </g>;
            })}
          </svg>
          <div className="flex items-center justify-bet
            <span>Шаг {dfsIdx+1}/{DFS_STEPS.length}</sp
            <div className="flex-1 mx-2 bg-slate-950 h-
              <div className="bg-indigo-600 h-full tran
            div>
            </div>
          </div>
        </div>

        {/* Code panel + stack */}
        <div className="md:col-span-2 space-y-3">
```

```
              }
            </div>
          </div>

          {/* Log */}
          <div className="bg-indigo-950/10 border borde
            <p className="text-[11px] text-slate-300 le
            <div className="mt-2 pt-2 border-t border-s
              <span className="text-slate-400">Мосты:</
              <span className="font-mono font-bold text
                {step.bridges.length > 0 ? step.bridges
              </span>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}
```

ФАЙЛ 33/74: src/components/DijkstraViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО

```
import { useState, useEffect } from 'react';

interface Node { id: string; x: number; y: number; name
interface Edge { u: string; v: string; w: number; }
interface Step {
  currentNode: string | null;
  checkingEdge: { u: string; v: string } | null;
  distances: { [key: string]: number };
  visited: { [key: string]: boolean };
  previous: { [key: string]: string | null };
  log: string;
  activeCodeLine: number;
```

```
    { line: 5, text: '        d, u = pq.pop() // Извлека
    { line: 6, text: '        if d > dist[u]: continue'
    { line: 7, text: '        for v, weight in graph.ne
    { line: 8, text: '            if dist[u] + weight <
    { line: 9, text: '                dist[v] = dist[u]
    { line: 10, text: '                pq.push((dist[v]
    { line: 11, text: '    return dist // Все кратчайши
  ];

  const [steps, setSteps] = useState<Step[]>([]);
  const [currentStepIndex, setCurrentStepIndex] = useSta
  const [isPlaying, setIsPlaying] = useState(false);

  useEffect(() => {
    const simulationSteps: Step[] = [];
    const dist: Record<string, number> = { A: 0, B: 999
    const visited: Record<string, boolean> = { A: false
    const prev: Record<string, string | null> = { A: nu

    simulationSteps.push({
      currentNode: null, checkingEdge: null,
      distances: { ...dist }, visited: { ...visited }, 
      log: '  Дейкстра готов к доставке. Начинаем из Пи
      activeCodeLine: 2,
    });

    for (let iter = 0; iter < nodes.length; iter++) {
      let minD = 999;
      let u = null;
      for (const n of nodes) {
        if (!visited[n.id] && dist[n.id] < minD) {
          minD = dist[n.id];
          u = n.id;
        }
      }

      if (!u) break;
```

```
      activeCodeLine: 11,
    });

    setSteps(simulationSteps);
  }, []);

  useEffect(() => {
    let timer: any = null;
    if (isPlaying && currentStepIndex < steps.length -
      timer = setTimeout(() => {
        setCurrentStepIndex((prev) => prev + 1);
      }, 1500);
    } else if (currentStepIndex >= steps.length - 1) {
      setIsPlaying(false);
    }
    return () => clearTimeout(timer);
  }, [isPlaying, currentStepIndex, steps.length]);

  const currentStep = steps[currentStepIndex] || null;

  if (!currentStep) return <div className="text-white">

  return (
    <div className="bg-slate-800/90 p-6 rounded-2xl bor
      {/* Шапка симулятора */}
      <div className="flex flex-wrap items-center justi
        <div className="flex items-center gap-3">
          <div className="p-2.5 bg-indigo-600 text-whit
            <span className="text-2xl"> </span>
          </div>
          <div>
            <h3 className="text-2xl font-bold text-whit
            <p className="text-sm text-indigo-300">Полн
          </div>
        </div>

        <div className="flex items-center gap-2">
          <button
```

```
              </marker>
              <marker id="arrow-dh-active" markerWidth=
                <path d="M0,0 L6,3 L0,6 z" fill="#f59e0
              </marker>
              <marker id="arrow-dh-opt" markerWidth="6"
                <path d="M0,0 L6,3 L0,6 z" fill="#10b98
              </marker>
            </defs>
            {/* Рёбра */}
            {edges.map((edge, idx) => {
              const uNode = nodes.find((n) => n.id === 
              const vNode = nodes.find((n) => n.id === 
              const isChecking =
                currentStep.checkingEdge &&
                (currentStep.checkingEdge.u === edge.u 

              const isOptimal = currentStep.previous[ed

              let marker = 'url(#arrow-dh-normal)';
              if (isChecking) marker = 'url(#arrow-dh-a
              else if (isOptimal) marker = 'url(#arrow-

              return (
                <g key={idx}>
                  <line
                    x1={uNode.x} y1={uNode.y}
                    x2={vNode.x} y2={vNode.y}
                    stroke={isChecking ? '#f59e0b' : is
                    strokeWidth={isChecking ? 5 : isOpt
                    markerEnd={marker}
                    className="transition-all duration-
                    strokeDasharray={isChecking ? '4,4'
                  />
                  <circle
                    cx={(uNode.x + vNode.x) / 2} cy={(u
                    r="12" fill="#1e293b"
                    stroke={isChecking ? '#f59e0b' : '#
                  />
```

```
              );
            })}
          </svg>
          <div className="w-full bg-slate-950/80 p-4 ro
            <p className="font-semibold text-amber-400 
            <p className="min-h-[40px] flex items-cente
          </div>
        </div>

        {/* Список смежности */}
        <div className="w-full lg:w-1/3 bg-emerald-950/
          <h4 className="text-lg font-bold text-emerald
          <div className="space-y-2.5 flex-1 overflow-y
            {Object.keys(adjList).map((u) => {
              const isCurrentNode = currentStep.current
              return (
                <div key={u} className={`p-3 rounded-lg
                    isCurrentNode ? 'bg-emerald-900/60 
            t-slate-300'
                  }`}>
                  <div className="flex items-center gap
                    <span className={`px-2 py-0.5 round
            te-300'}`}>
                      {u}
                    </span>
                    <span className="text-slate-400">→<
                  </div>
                  <div className="flex flex-wrap gap-2 
                    {adjList[u].map((edge, idx) => {
                      const isCheckingThis = currentSte
                      return (
                        <span key={idx} className={`px-
                            isCheckingThis ? 'bg-amber-
            600 text-slate-300'
                          }`}>
                          <span>{edge.v}</span><span cl
                          <span className={isCheckingTh
            an>
```

```
                          {n.name}
                        </td>
                        <td className="py-2.5 px-3 font
                          {dist === 999 ? '∞' : dist}
                        </td>
                        <td className="py-2.5 px-3 text
                          {prev || '–'}
                        </td>
                      </tr>
                    );
                  }))}
                </tbody>
              </table>
            </div>
          </div>
        </div>

        {/* Код алгоритма */}
        <div className="w-full lg:w-1/2 bg-slate-900/60
          <div>
            <h4 className="text-lg font-bold text-slate
            <div className="bg-slate-950/80 rounded-lg p
              {codeLines.map((item) => {
                const isActive = currentStep.activeCode
                return (
                  <div key={item.line} className={`py-1
                      isActive ? 'bg-indigo-900/80 text
                    }`}>
                    <span className="text-slate-600 sel
                    <span className="whitespace-pre fle
                    {isActive && <span className="text-
            </span>}
                  </div>
                );
              }))}
            </div>
          </div>
        </div>
```

```
  const [activeTab, setActiveTab] = useState<'table' |

  useEffect(() => {
    const generatedSteps: DPStep[] = [];
    let stepId = 0;
    const memoMap: { [key: number]: number } = {};

    function simulateFib(n: number): number {
      generatedSteps.push({
        id: stepId++,
        n,
        action: 'call',
        desc: `Вызов fibMemo(${n}). Проверяем наличие в
        codeLine: 2,
        memoState: { ...memoMap }
      });

      if (n in memoMap) {
        generatedSteps.push({
          id: stepId++,
          n,
          action: 'memo_hit',
          desc: `⚡ Кэш-хит! fibMemo(${n}) уже посчитано
          codeLine: 2,
          memoState: { ...memoMap }
        });
        return memoMap[n];
      }

      if (n <= 2) {
        generatedSteps.push({
          id: stepId++,
          n,
          action: 'base_case',
          desc: `Базовый случай: n <= 2 (n=${n}). Возвра
          codeLine: 3,
          memoState: { ...memoMap }
        });
```

```
    setSteps(generatedSteps);
    setCurrentStepIndex(0);
    setIsPlaying(false);
  }, [targetN]);

  useEffect(() => {
    let timer: any = null;
    if (isPlaying && currentStepIndex < steps.length -
      timer = setTimeout(() => {
        setCurrentStepIndex(prev => prev + 1);
      }, speed);
    } else if (currentStepIndex >= steps.length - 1) {
      setIsPlaying(false);
    }
    return () => clearTimeout(timer);
  }, [isPlaying, currentStepIndex, steps, speed]);

  const currentStep = steps[currentStepIndex] || null;
  const memoState = currentStep?.memoState || {};

  return (
    <div className="bg-slate-900 rounded-2xl border bor
      {/* Header & Controls */}
      <div className="flex flex-col lg:flex-row lg:item
        <div>
          <div className="flex items-center gap-3 mb-2"
            <div className="p-2 bg-emerald-500/10 borde
              <RefreshCw className="w-6 h-6" />
            </div>
            <h3 className="text-2xl font-extrabold text
          </div>
          <p className="text-sm text-slate-400">
            Смотри, как таблица DP кэширует результаты
          </p>
        </div>

        <div className="flex flex-wrap items-center gap
```

```
            >
              {isPlaying ? <Pause className="w-4 h-4" /
              {isPlaying ? 'Пауза' : 'Пуск'}
            </button>
            <button
              onClick={() => { setIsPlaying(false); set
              disabled={currentStepIndex === steps.leng
              className="p-2 text-slate-400 hover:text-
              title="Шаг вперед"
            >
              <ChevronRight className="w-5 h-5" />
            </button>
          </div>
        </div>
      </div>

      {/* DP Table View */}
      <div className="mb-8 bg-slate-950 p-4 md:p-6 roun
        <h4 className="text-xs font-bold uppercase trac
        <div className="flex flex-wrap items-center gap
          {Array.from({ length: targetN }, (_, i) => i +
            const isCached = num in memoState || num <=
            const val = num <= 2 ? 1 : memoState[num];
            const isCurrent = currentStep?.n === num;

            return (
              <div
                key={num}
                className={`flex flex-col items-center
                  isCurrent
                    ? 'bg-emerald-950 border-emerald-50
                    : isCached
                    ? 'bg-slate-900 border-emerald-500/
                    : 'bg-slate-900/50 border-slate-800
                }`}
              >
                <span className="text-xs text-slate-400
                <span className={`text-xl md:text-2xl f
```

```
              ? 'border-emerald-500 text-emerald-400 bg
              : 'border-transparent text-slate-400 hove
          }`}
        >
          <Code className="w-4 h-4" />
          Интерактивный код
        </button>
      </div>

      {/* Tab Content */}
      <div className="min-h-[320px] flex flex-col justi
        {activeTab === 'table' && (
          <div className="bg-slate-950 p-6 rounded-2xl i
            <h5 className="font-bold text-white text-sm
              <span className="text-emerald-400"> </spa
            </h5>
            <p className="text-sm text-slate-400">
              В классической рекурсии для N={targetN} п
             вычисленное значение в таблице, мы момента
            </p>
            <div className="p-4 bg-slate-900 rounded-xl
            t-slate-300">
              <span>Всего шагов симуляции: {steps.lengt
              <span className="text-emerald-400 font-bo
            </div>
          </div>
        )}

        {activeTab === 'code' && (
          <div className="bg-slate-950 rounded-2xl bord
            <div className="bg-slate-900 px-6 py-3 bord
              <span className="text-xs font-bold font-m
              <span className="text-xs text-emerald-400
            </div>
            <pre className="p-6 font-mono text-sm space
              {DP_CODE_LINES.map((line, idx) => {
                const lineNum = idx + 1;
                const isActive = currentStep?.codeLine =
```

```
----------------------------------------------------------------
}


----------------------------------------------------------------
ФАЙЛ 35/74: src/components/EulerSimulator.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
----------------------------------------------------------------

import { useState } from "react";
import { Undo } from "lucide-react";

const C1_NODES = [
  { id: 0, label: "A", x: 190, y: 60 },
  { id: 1, label: "B", x: 280, y: 140 },
  { id: 2, label: "C", x: 100, y: 140 },
  { id: 3, label: "D", x: 280, y: 250 },
  { id: 4, label: "E", x: 100, y: 250 }
];

const C1_EDGES = [
  { u: 0, v: 1 }, { u: 0, v: 2 },
  { u: 1, v: 3 }, { u: 2, v: 4 },
  { u: 1, v: 2 },
  { u: 3, v: 4 }
];

export function EulerSimulator() {
  const [c1Path, setC1Path] = useState<number[]>([]);
  const [c1VisitedEdges, setC1VisitedEdges] = useState<
  const [c1Msg, setC1Msg] = useState("Кликни на вершину
  const [c1Done, setC1Done] = useState(false);

  const resetC1 = () => {
    setC1Path([]); setC1VisitedEdges([]); setC1Msg("Кли
  };

  const clickC1 = (vId: number) => {
    if (c1Done && c1Path.length > 0) return;
```

```
      <div className="flex items-center justify-between
        <h4 className="text-base font-bold text-white">
        <button onClick={resetC1} className="flex items
          old border border-slate-700 transition-all">
          <Undo className="h-3.5 w-3.5" /> Сброс
        </button>
      </div>

      <div className="bg-slate-950 rounded-xl p-4 borde
        -hidden">
        <div className="absolute inset-0 bg-[radial-gra
        <svg className="w-full h-[260px] z-10 relative"
          {C1_EDGES.map((e, i) => {
            const u = C1_NODES.find(n => n.id === e.u)!
            const v = C1_NODES.find(n => n.id === e.v)!
            const key = `${Math.min(e.u,e.v)}-${Math.ma
            const used = c1VisitedEdges.includes(key);
            return <line key={i} x1={u.x} y1={u.y} x2={
              stroke={used ? "#6366f1" : "#475569"}
              strokeWidth={used ? 4 : 2}
              className="transition-all duration-300" /
          })}
          {C1_NODES.map(n => {
            const isLast = c1Path[c1Path.length-1] === 
            const visited = c1Path.includes(n.id);
            return <g key={n.id} className="cursor-poin
              {isLast && <circle cx={n.x} cy={n.y} r={1
              <circle cx={n.x} cy={n.y} r={11}
                fill={isLast ? "#6366f1" : visited ? "#
                stroke={isLast ? "#fff" : visited ? "#6
                strokeWidth={2.5}
                className="transition-all duration-200 
              <text x={n.x} y={n.y+4} textAnchor="middl
            bel}</text>
            </g>;
          })}
        </svg>
        <div className="absolute top-2 left-2 bg-slate-
```

```
      ер, красный) цвет?',
    options: [
      'Мы радуемся и добавляем его в минимальное остовн
      'Мы перекрашиваем их в синий цвет и делим граф по
      'Мы безжалостно ВЫБРАСЫВАЕМ это ребро, иначе полу
      'Мы вызываем алгоритм Дейкстры для поиска нового р
    ],
    correctAnswer: 2,
    explanation: 'Верно! Если вершины уже в одном клане
      Это нарушит структуру дерева.'
  },
  {
    id: 2,
    question: 'Почему алгоритм Дейкстры впадает в ступор
    options: [
      'Потому что он написан только для положительных ч
      'Он жадный и считает, что каждый следующий шаг де
         закрытые вершины.',
      'Отрицательные ребра создают гравитационный колла
      'Дейкстра просто не любил отрицательные балансы на
    ],
    correctAnswer: 1,
    explanation: 'Именно! Дейкстра уверен: если ты прие
      ля чего нужен алгоритм Беллмана-Форда.'
  },
  {
    id: 3,
    question: 'Какова ключевая фишка алгоритма Борувки
    options: [
      'Он использует меньше памяти благодаря коммунисти
      'Он умеет работать с отрицательными весами без ци
      'Он позволяет выполнять шаги слияния колхозов (ко
      'Он был разработан в Токио и поэтому самый быстры
    ],
    correctAnswer: 2,
    explanation: 'Абсолютно верно! В Борувке каждая ком
      лелить вычисления на GPU или многопроцессорных кл
  },
```

```
    setSelectedOption(index);
    setIsAnswered(true);
    if (index === currentQ.correctAnswer) {
      setScore(prev => prev + 1);
    }
  };

  const handleNext = () => {
    if (currentQIndex + 1 < quizQuestions.length) {
      setCurrentQIndex(prev => prev + 1);
      setSelectedOption(null);
      setIsAnswered(false);
    } else {
      setShowResults(true);
    }
  };

  const handleRestart = () => {
    setCurrentQIndex(0);
    setSelectedOption(null);
    setIsAnswered(false);
    setScore(0);
    setShowResults(false);
  };

  if (showResults) {
    return (
      <div className="bg-slate-900/90 border border-sla
        my-12">
        <div className="w-20 h-20 bg-gradient-to-tr fro
          -6 shadow-xl shadow-indigo-500/30">
          <Award className="w-10 h-10 text-white" />
        </div>
        <h3 className="text-3xl font-bold text-white mb
        <p className="text-slate-400 text-sm mb-6">Ты п

        <div className="bg-slate-950 border border-slat
          <p className="text-slate-400 text-sm uppercas
```

```
        <h4 className="text-xl md:text-2xl font-bold te
          {currentQ.question}
        </h4>

        <div className="space-y-4">
          {currentQ.options.map((option, idx) => {
            const isSelected = selectedOption === idx;
            const isCorrect = idx === currentQ.correctA

            let optionStyle = "bg-slate-800/80 hover:bg
            if (isAnswered) {
              if (isCorrect) {
                optionStyle = "bg-emerald-950/80 border
              } else if (isSelected) {
                optionStyle = "bg-rose-950/80 border-ro
              } else {
                optionStyle = "bg-slate-800/40 border-s
              }
            }

            return (
              <div
                key={idx}
                onClick={() => handleSelectOption(idx)}
                className={"flex items-start gap-4 p-4 
              >
                <div className="mt-0.5 shrink-0">
                  {isAnswered && isCorrect ? (
                    <CheckCircle2 className="w-6 h-6 te
                  ) : isAnswered && isSelected && !isCo
                    <XCircle className="w-6 h-6 text-ro
                  ) : (
                    <div className={"w-6 h-6 rounded-fu
            order-indigo-500 bg-indigo-500 text-white"
                      {String.fromCharCode(65 + idx)}
                    </div>
                  )}
                </div>
```

ФАЙЛ 37/74: src/components/FloydViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО

```
import { useState, useEffect } from 'react';

interface Step {
  k: number; i: number; j: number;
  matrix: number[][]; updated: boolean;
  log: string; activeCodeLine: number;
}

export default function FloydViz() {
  const [steps, setSteps] = useState<Step[]>([]);
  const [currentStepIndex, setCurrentStepIndex] = useSta
  const [isPlaying, setIsPlaying] = useState(false);

  const nodeNames = ['А 0', 'Б 1', 'В 2', 'Г 3', 'Д 4',
  const nodesSvg = [
    { id: 0, name: 'Аэропорт 0', x: 100, y: 60 },
    { id: 1, name: 'Аэропорт 1', x: 250, y: 60 },
    { id: 2, name: 'Аэропорт 2', x: 400, y: 60 },
    { id: 3, name: 'Аэропорт 3', x: 100, y: 180 },
    { id: 4, name: 'Аэропорт 4', x: 400, y: 180 },
    { id: 5, name: 'Аэропорт 5', x: 175, y: 280 },
    { id: 6, name: 'Аэропорт 6', x: 325, y: 280 },
  ];

  const initialMatrix = [
    [0, 4, 999, 2, 999, 999, 999],
    [999, 0, 3, 999, 999, 3, 999],
    [999, 999, 0, 999, 2, 999, 999],
    [999, 999, 999, 0, 4, 2, 999],
    [999, 999, 999, 999, 0, 999, 1],
    [999, 999, 999, 999, 999, 0, 5],
```

```
        log: `→  Внешний цикл. Разрешаем пути через про
        activeCodeLine: 3,
      });

      for (let i = 0; i < N; i++) {
        for (let j = 0; j < N; j++) {
          if (i === j || i === k || j === k) continue;

          const oldVal = dist[i][j];
          const viaVal = dist[i][k] + dist[k][j];
          if (dist[i][k] !== 999 && dist[k][j] !== 999 
            dist[i][j] = viaVal;
            simulationSteps.push({
              k, i, j, matrix: dist.map(r => [...r]), u
              log: `  Улучшение пути [${i}]→[${j}] чере
            ∞'}.`,
              activeCodeLine: 7,
            });
          }
        }
      }
    }

    simulationSteps.push({
      k: 7, i: -1, j: -1, matrix: dist.map(r => [...r])
      log: '  Все пары отработаны. Алгоритм Флойда заве
      activeCodeLine: 8,
    });

    setSteps(simulationSteps);
    setCurrentStepIndex(0);
    setIsPlaying(false);
  }, []);

  useEffect(() => {
    let timer: any = null;
    if (isPlaying && currentStepIndex < steps.length -
      timer = setTimeout(() => setCurrentStepIndex((p) =
```

```
            {currentStep.k >= 0 && currentStep.k < 7 ?
          </div>

          <svg className="w-full h-auto max-h-[500px]" 
            <defs>
              <marker id="arrow-fw-normal" markerWidth=
            <path d="M0,0 L6,3 L0,6 z" fill="#475569" /
              <marker id="arrow-fw-active" markerWidth=
            <path d="M0,0 L6,3 L0,6 z" fill="#10b981" /
              <marker id="arrow-fw-via" markerWidth="6"
            th d="M0,0 L6,3 L0,6 z" fill="#818cf8" /></
            </defs>
            {Object.keys(adjList).flatMap((uStr) => {
              const u = parseInt(uStr);
              return adjList[u].map((edge) => {
                const uNode = nodesSvg.find((n) => n.id
                const vNode = nodesSvg.find((n) => n.id
                const isTarget = currentStep.i === u &&
                const isVia = (currentStep.i === u && c
                let strokeColor = '#475569'; let marker
                if (isTarget) { strokeColor = '#10b981'
                else if (isVia) { strokeColor = '#818cf
                return (
                  <g key={`${u}-${edge.v}`}>
                    <line x1={uNode.x} y1={uNode.y} x2=
             markerEnd={marker} strokeDasharray={isTarg
                    <circle cx={(uNode.x + vNode.x) / 2
             isVia ? '#818cf8' : '#64748b'} strokeWidth=
                    <text x={(uNode.x + vNode.x) / 2} y=
            8fafc'} fontSize="10" fontWeight="bold" tex
                  </g>
                );
              });
            })}
            {nodesSvg.map((node) => {
              const isK = currentStep.k === node.id;
              const isI = currentStep.i === node.id;
              const isJ = currentStep.j === node.id;
```

```
                    const isUpd = currentStep.i === u
                    const isVia = currentStep.k !== -
          rentStep.j === edge.v);
                    return (
                      <span key={idx} className={`px-
          dow' : isVia ? 'bg-indigo-500 text-white bo
                        {edge.v} ({edge.w})
                      </span>
                    )
                  })}
                </div>
              );
            })}
          </div>
        </div>
      </div>

      <div className="flex flex-col lg:flex-row gap-6">
        {/* Матрица расстояний */}
        <div className="w-full lg:w-1/2 bg-rose-950/10
          <div className="flex justify-between items-ce
            <h4 className="text-lg font-bold text-rose-
          </div>
          <div className="overflow-x-auto">
            <table className="w-full text-center border
              <thead><tr className="bg-rose-950/40 text
                <th className="p-2 border-r border-rose
                {nodeNames.map((n, idx) => <th key={idx
          -400 font-bold' : ''}`}>{n.split(' ')[0]} {
              </tr></thead>
              <tbody className="text-xs font-mono">
                {currentStep.matrix.map((row, i) => (
                  <tr key={i} className="border-b borde
                    <td className={`p-2 bg-rose-950/40
          tStep.k === i ? 'text-amber-400' : ''}`}>
                        {nodeNames[i].split(' ')[0]} {i}
                    </td>
                    {row.map((val, j) => {
```

ФАЙЛ 38/74: src/components/GraphTraversalViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО

```
import { useState, useEffect, useMemo } from 'react';
import { Play, Pause, SkipForward, Undo, RefreshCw } fr

type Node = { id: string; label: string; x: number; y: 
type Edge = { u: string; v: string };

const nodes: Node[] = [
    { id: 'A', label: 'A', x: 200, y: 40 },
    { id: 'B', label: 'B', x: 100, y: 120 },
    { id: 'C', label: 'C', x: 300, y: 120 },
    { id: 'D', label: 'D', x: 50, y: 200 },
    { id: 'E', label: 'E', x: 150, y: 200 },
    { id: 'F', label: 'F', x: 250, y: 200 },
    { id: 'G', label: 'G', x: 350, y: 200 },
];

const edges: Edge[] = [
    { u: 'A', v: 'B' }, { u: 'A', v: 'C' },
    { u: 'B', v: 'D' }, { u: 'B', v: 'E' },
    { u: 'C', v: 'F' }, { u: 'C', v: 'G' },
];

const adj: Record<string, string[]> = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F', 'G'],
    'D': ['B'],
    'E': ['B'],
    'F': ['C'],
    'G': ['C'],
};

type Action = {
```

```
    const q: string[] = [];

    q.push(start);
    vis.add(start);
    steps.push({ type: 'visit', node: start, desc: `Нач
      ray.from(vis)] });

    while (q.length > 0) {
        const v = q[0];
        steps.push({ type: 'process', node: v, desc: `Д
          q], visitedNodes: [...Array.from(vis)] });
        q.shift();

        for (const u of adj[v]) {
            if (!vis.has(u)) {
                vis.add(u);
                q.push(u);
                steps.push({ type: 'visit', node: u, de
            tack: [...q], visitedNodes: [...Array.from(
            }
        }
    }

    steps.push({ type: 'backtrack', node: '', desc: `Оч
      om(vis)] });

    return steps;
}


export function GraphTraversalViz() {
    const [mode, setMode] = useState<"dfs" | "bfs">("df
    const [autoPlay, setAutoPlay] = useState(false);
    const [stepIdx, setStepIdx] = useState(0);

    const steps = useMemo(() => mode === 'dfs' ? genera

    useEffect(() => {
```

```
            <div className="flex flex-col md:flex-row ga
                {/* SVG Graph View */}
                <div className="bg-slate-900 border bord
            ive min-h-[300px]">
                     <svg className="w-full h-full max-w
                        {/* Edges */}
                        {edges.map((e, i) => {
                            const u = nodes.find(n => n
                            const v = nodes.find(n => n
                            // Check if edge is part of
                            const edgeTraversed = isVis

                            return (
                                <line key={i} x1={u.x} y
                                      className={`strok
            500' : 'stroke-emerald-500') : 'stroke-slate
                            );
                        })}

                        {/* Nodes */}
                        {nodes.map(n => {
                            const visited = isVisited(n
                            const current = isCurrent(n

                            let fillColor = "fill-slate
                            let strokeColor = "stroke-s
                            let textColor = "fill-slate
                            let radius = 18;

                            if (visited) {
                                fillColor = mode === 'd
                                strokeColor = mode ===
                                textColor = "fill-white
                            }

                            if (current && step.type !==
                                strokeColor = "stroke-a
```

```
                                ${idx === (mode === 'df
            0_10px_rgba(245,158,11,0.2)]' : ''}
                            `}>
                                Узел: {item}
                                {idx === (mode === 'dfs'
                                    <span className="ml-
                                )}
                            </div>
                        ))}
                        {(!step.queueOrStack || step.que
                            <div className="text-slate-5
                        )}
                    </div>
                </div>
            </div>

            <div className="flex flex-col sm:flex-row g
                {/* Controls */}
                <div className="flex bg-slate-900 borde
                    <button onClick={() => {setAutoPlay
            "p-2 text-slate-400 hover:text-white hover:
                        <Undo strokeWidth={3} size={16}
                    </button>
                    <button onClick={() => setAutoPlay(
             justify-center min-w-[80px] ${mode === 'df
            >
                        {autoPlay ? <><Pause size={16} 
                    </button>
                    <button onClick={() => {setAutoPlay
            = steps.length - 1} className="p-2 text-sla
            hover:bg-transparent">
                        <SkipForward strokeWidth={3} si
                    </button>
                    <button onClick={() => {setAutoPlay
            g-slate-800 rounded-lg ml-2 border-l border
                        <RefreshCw strokeWidth={3} size
                    </button>
                </div>
```

```
    const parent = Math.floor((idx - 1) / 2);
    if (a[parent].priority < a[idx].priority) {
      [a[parent], a[idx]] = [a[idx], a[parent]];
      idx = parent;
    } else break;
  }
  return a;
}

function siftDown(arr: Patient[]): Patient[] {
  const a = arr.map(x => ({ ...x }));
  const n = a.length;
  let idx = 0;
  while (true) {
    let largest = idx;
    const l = 2 * idx + 1;
    const r = 2 * idx + 2;
    if (l < n && a[l].priority > a[largest].priority) la
    if (r < n && a[r].priority > a[largest].priority) la
    if (largest !== idx) {
      [a[largest], a[idx]] = [a[idx], a[largest]];
      idx = largest;
    } else break;
  }
  return a;
}

function heapInsert(arr: Patient[], item: Patient): Pat
  return siftUp([...arr, { ...item }]);
}

// Position in tree layout
function nodePos(index: number): { x: number; y: number
  const level = Math.floor(Math.log2(index + 1));
  const posInLevel = index - (Math.pow(2, level) - 1);
  const count = Math.pow(2, level);
  const x = ((posInLevel + 0.5) / count) * 100;
  const y = 10 + level * 24;
```

```
  const [healingPatient, setHealing]    = useState<Pati

  const addLog = (msg: string) => setLog(prev => [msg,

  // --- INSERT: Новый пациент поступает в приемный пок
  const insertPatient = useCallback((name: string, prio
    if (busy || heap.length >= 15) {
      addLog('⚠ Все палаты переполнены! Дождитесь выпис
      setShake(true);
      setTimeout(() => setShake(false), 400);
      return;
    }
    setBusy(true);

    const preset = PATIENT_PRESETS[Math.floor(Math.rand
    const finalName = name || preset.name.split(' ')[0]
    const finalEmoji = preset.emoji;
    const finalSymptom = preset.symptom;

    const newPatient: Patient = {
      id: `p${uid++}`,
      name: finalName,
      emoji: finalEmoji,
      symptom: finalSymptom,
      priority,
      animState: 'in',
    };

    const nextHeap = heapInsert(heap, newPatient).map(x
    // Mark specifically this new patient as 'in' in th
    const target = nextHeap.find(x => x.priority === pr
    const marked = nextHeap.map(x => x.id === (target?.

    setHeap(marked);
    addLog(`  Поступил пациент: ${finalName} с тяжестью

    setTimeout(() => {
      setHeap(nextHeap);
```

```
      setTimeout(() => {
        // Run binary heap extraction
        setHeap(prev => {
          if (prev.length === 0) return [];
          const a = prev.map(x => ({ ...x }));
          a[0] = { ...a[a.length - 1] };
          a.pop();
          return siftDown(a).map(x => ({ ...x, animStat
        });

        // Patient gets healed in the bed
        setTimeout(() => {
          setHealing(prev => prev ? { ...prev, animStat
          addLog(`  Успех! Пациент ${topPatient.name} п
          setTimeout(() => {
            setHealing(null);
            setBusy(false);
          }, 600);
        }, 800);

      }, 350);
    }, 650);
  }, [busy, heap]);

  const reset = () => {
    setHeap(INITIAL_PATIENTS.map(x => ({ ...x, animStat
    setLog(['  База данных скорой помощи перезагружена.
    setHealing(null);
    setTimeout(() => setHeap(INITIAL_PATIENTS.map(x =>
  };

  // --- Render binary tree lines ---
  const edges = heap.flatMap((_, idx) => {
    const res: React.ReactElement[] = [];
    const p = nodePos(idx);
    const l = 2 * idx + 1;
    const r = 2 * idx + 2;
```

```
      </div>

      {/* УПРАВЛЕНИЕ */}
      <div className="flex flex-col lg:flex-row items-s

        {/* Добавить случайного */}
        <button
          onClick={handleRandomInsert}
          disabled={busy || heap.length >= 15}
          className="flex-1 min-w-[150px] bg-gradient-t
            opacity-40 disabled:cursor-not-allowed text
            ve:scale-95 transition-all cursor-pointer f
        >
          <span>  Привезти по Скорой (INSERT)</span>
        </button>

        {/* Добавить кастомного */}
        <form onSubmit={handleCustomInsert} className="
          <input
            type="text"
            required
            value={customName}
            onChange={e => setCustomName(e.target.value
            placeholder="Имя пациента"
            className="flex-1 bg-slate-800 border borde
            er-emerald-500 transition-colors placeholde
          />
          <div className="flex flex-col items-center mi
            <span className="text-[9px] font-mono text-
            <input
              type="range"
              min="10"
              max="99"
              value={customPriority}
              onChange={e => setCustomPriority(Number(e
              className="w-20 accent-emerald-500 cursor
            />
          </div>
```

```
          </div>

          <div className="relative w-full h-full min-h-
            <svg className="absolute inset-0 w-full h-f
              {edges}
            </svg>

            {heap.map((patient, idx) => {
              const pos = nodePos(idx);
              const isRoot = idx === 0;
              return (
                <div
                  key={patient.id}
                  className={`
                    absolute flex flex-col items-center
                    -translate-x-1/2 -translate-y-1/2 t
                    ${patient.animState === 'in' ? 'ani
                    ${patient.animState === 'winner' ?
                    ${patient.animState === 'out' ? 'an
                    ${isRoot ? 'bg-rose-950/80 border-r
                  `}
                  style={{
                    left: `${pos.x}%`,
                    top: `${pos.y}%`,
                    width: isRoot ? 60 : 52,
                    height: isRoot ? 60 : 52,
                  }}
                >
                  <span className="text-2xl leading-non
                  <span className="text-[10px] font-mon
                    {patient.priority}%
                  </span>

                  {/* Tooltip */}
                  <div className="absolute bottom-full
                    <div className="bg-slate-950 border
            0 shadow-xl">
                      <div className="font-bold text-wh
```

```
---------------------------------------------------------------
            ) : (
              <div className="text-center text-slate-60
                <span className="text-5xl mb-2 block"> 
                <p className="text-xs font-bold font-mo
                <p className="text-[10px] mt-1">Ждём ca
              </div>
            )}
          </div>

          <div className="w-full h-3 bg-gradient-to-r f
        </div>

      </div>

      {/* ЛОГ ДЕЙСТВИЙ */}
      <div className="bg-slate-900 border border-slate-
        <div className="text-[10px] font-mono text-slat
        {log.map((entry, idx) => (
          <div
            key={idx}
            className={`text-xs font-mono flex items-ce
          >
            {idx === 0 ? <span className="text-emerald-
            <span>{entry}</span>
          </div>
        ))}
      </div>
    </div>
  );
};


---------------------------------------------------------------
ФАЙЛ 40/74: src/components/hints/demoKit.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
---------------------------------------------------------------

import React, { useEffect, useState } from "react";
```

```
  y: number;
  r?: number;
  fill: string;
  label?: string | number;
  stroke?: string;
}> = ({ x, y, r = 13, fill, label, stroke }) => (
  <g style={{ transition: "all .35s ease" }}>
    <circle cx={x} cy={y} r={r} fill={fill} stroke={str
    {label !== undefined && (
      <text x={x} y={y} textAnchor="middle" dominantBas
        {label}
      </text>
    )}
  </g>
);

export const Edge: React.FC<{
  a: [number, number];
  b: [number, number];
  color?: string;
  width?: number;
  dashed?: boolean;
}> = ({ a, b, color = C.edge, width = 2, dashed }) => (
  <line
    x1={a[0]}
    y1={a[1]}
    x2={b[0]}
    y2={b[1]}
    stroke={color}
    strokeWidth={width}
    strokeDasharray={dashed ? "4 3" : undefined}
    style={{ transition: "stroke .35s ease" }}
  />
);
```

ФАЙЛ 41/74: src/components/hints/demosGraph.tsx

```
    <Svg caption={head ? `Идём вглубь: ${visited.join("
      {GRAPH_EDGES.map(([u, v], i) => {
        const iu = visited.indexOf(u);
        const iv = visited.indexOf(v);
        const onPath = iu >= 0 && iv >= 0 && Math.abs(i
        return <Edge key={i} a={GRAPH_POS[u]} b={GRAPH_
      })}
      {Object.entries(GRAPH_POS).map(([k, [x, y]]) => (
        <Node key={k} x={x} y={y} label={k} fill={k ===
      ))}
    </Svg>
  );
};

export const ToposortDemo: React.FC = () => {
  const nodes = [
    { id: "трусы", x: 52, y: 32 }, { id: "штаны", x: 14
    { id: "носки", x: 52, y: 98 }, { id: "боты", x: 148
  ];
  const edges: [string, string][] = [["трусы", "штаны"]
  const order = ["трусы", "носки", "штаны", "боты"];
  const step = useLoop(order.length + 1, 800);
  const placed = order.slice(0, step);
  const pos = Object.fromEntries(nodes.map((n) => [n.id
  return (
    <Svg caption={`Порядок: ${placed.join(" → ") || "…"
      {edges.map(([u, v], i) => (
        <Edge key={i} a={pos[u]} b={pos[v]} color={plac
      ))}
      {nodes.map((n) => {
        const idx = placed.indexOf(n.id);
        return (
          <g key={n.id}>
            <rect x={n.x - 34} y={n.y - 12} width={68} 
            " }} />
            <text x={n.x} y={n.y} textAnchor="middle" d
            {idx >= 0 && <circle cx={n.x + 30} cy={n.y
            {idx >= 0 && (
```

```
  return (
    <Svg caption={cut ? "Убрали C–D → граф развалился на
      {edges.map(([u, v], i) => {
        const isBridge = u === "C" && v === "D";
        if (isBridge && cut) return null;
        return <Edge key={i} a={pos[u]} b={pos[v]} colo
      })}
      {Object.entries(pos).map(([k, [x, y]]) => (
        <Node key={k} x={x} y={y} label={k} fill={cut ?
      ))}
    </Svg>
  );
};

export const ArticulationDemo: React.FC = () => {
  const step = useLoop(2, 1300);
  const pos: Record<string, [number, number]> = {
    A: [36, 40], B: [36, 96], C: [130, 68], D: [224, 40
  };
  const edges: [string, string][] = [["A", "B"], ["A",
  const cut = step === 1;
  return (
    <Svg caption={cut ? "Убрали вершину C → две отдельн
      {edges.map(([u, v], i) => (cut && (u === "C" || v
      {Object.entries(pos).map(([k, [x, y]]) =>
        cut && k === "C" ? (
          <circle key={k} cx={x} cy={y} r={12} fill="no
        ) : (
          <Node key={k} x={x} y={y} label={k} fill={k ==
            } />
        )
      )}
    </Svg>
  );
};

export const MstDemo: React.FC = () => {
  const pos: Record<string, [number, number]> = {
```

```
      ))}
      {Object.entries(pos).map(([k, [x, y]]) => (
        <Node key={k} x={x} y={y} label={k} fill={step =
      ))}
    </Svg>
  );
};
```

ФАЙЛ 42/74: src/components/hints/demosStruct.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО

```
import React from "react";
import { C, Edge, Node, Svg, useLoop } from "./demoKit"

export const HeapDemo: React.FC = () => {
  const states = [[4, 8, 9, 12, 2], [4, 2, 9, 12, 8], [
  const step = useLoop(states.length, 1000);
  const heap = states[step];
  const pos: [number, number][] = [[130, 24], [80, 68],
  const moving = step === 0 ? 4 : step === 1 ? 1 : 0;
  return (
    <Svg caption="Новый элемент всплывает наверх, пока
      <Edge a={pos[0]} b={pos[1]} /><Edge a={pos[0]} b=
      <Edge a={pos[1]} b={pos[3]} /><Edge a={pos[1]} b=
      {heap.map((v, i) => (
        <Node key={i} x={pos[i][0]} y={pos[i][1]} label=
      ))}
    </Svg>
  );
};

export const StackDemo: React.FC = () => {
  const states = [["A"], ["A", "B"], ["A", "B", "C"], [
  const step = useLoop(states.length, 800);
  const items = states[step];
```

```
  );
};

export const DsuDemo: React.FC = () => {
  const parents = [[0, 1, 2, 3], [0, 0, 2, 3], [0, 0, 2
  const step = useLoop(parents.length, 950);
  const p = parents[step];
  const pos: [number, number][] = [[50, 95], [105, 95],
  const rootColor = ["#6366f1", "#10b981", "#f59e0b", "
  const root = (i: number): number => (p[i] === i ? i :
  return (
    <Svg caption="find = «кто твой староста», union = o
      {p.map((par, i) =>
        par !== i ? (
          <path key={i} d={`M ${pos[i][0]} ${pos[i][1]
            fill="none" stroke={rootColor[root(i)]} str
        ) : null
      )}
      {pos.map(([x, y], i) => (<Node key={i} x={x} y={y
    </Svg>
  );
};

const TRIE_NODES = [
  { id: 0, x: 130, y: 22, ch: "•", parent: null as numb
  { id: 1, x: 72, y: 62, ch: "a", parent: 0 },
  { id: 2, x: 188, y: 62, ch: "b", parent: 0 },
  { id: 3, x: 42, y: 104, ch: "b", parent: 1 },
  { id: 4, x: 102, y: 104, ch: "c", parent: 1 },
  { id: 5, x: 188, y: 104, ch: "c", parent: 2 },
];

export const TrieDemo: React.FC = () => {
  const step = useLoop(TRIE_NODES.length + 1, 700);
  return (
    <Svg caption="Путь от корня по буквам = слово; общи
      {TRIE_NODES.filter((n) => n.parent !== null).map(
        const p = TRIE_NODES[n.parent as number];
```

```
---------------------------------------------------------------
        const b = TRIE_NODES[to];
        return (
          <path key={i} d={`M ${a.x} ${a.y} Q ${(a.x + 
            fill="none" stroke={C.warn} strokeWidth={1.
        );
      }))}
      {TRIE_NODES.map((n) => (
        <Node key={n.id} x={n.x} y={n.y} label={n.ch} f
      ))}
    </Svg>
  );
};

export const SegmentTreeDemo: React.FC = () => {
  const step = useLoop(3, 1000);
  const nodes = [
    { x: 130, y: 24, w: 226, label: "[0..7]", lvl: 0 },
    { x: 72, y: 60, w: 110, label: "[0..3]", lvl: 1 },
    { x: 188, y: 60, w: 110, label: "[4..7]", lvl: 1 },
    { x: 44, y: 96, w: 52, label: "[0..1]", lvl: 2 },
    { x: 100, y: 96, w: 52, label: "[2..3]", lvl: 2 },
    { x: 160, y: 96, w: 52, label: "[4..5]", lvl: 2 },
    { x: 216, y: 96, w: 52, label: "[6..7]", lvl: 2 },
  ];
  return (
    <Svg caption="Запрос собирается из O(log n) готовых
      {nodes.map((n, i) => (
        <g key={i}>
          <rect x={n.x - n.w / 2} y={n.y - 10} width={n
            fill={n.lvl === step ? C.active : C.idle} s
          <text x={n.x} y={n.y} textAnchor="middle" dom
        </g>
      ))}
    </Svg>
  );
};

export const PrefixSumDemo: React.FC = () => {
```

```
        <g key={i}>
          <rect x={12 + i * 30} y={18} width={26} heigh
          <text x={25 + i * 30} y={29} textAnchor="midd
        </g>
      ))}
      {Array.from({ length: count }).map((_, i) => (
        <rect key={i} x={12 + i * 30} y={52 + (i % 4) *
          fill={C.active} opacity={i === 0 ? 1 : 0.4} s
      ))}
    </Svg>
  );
};

export const AmortizedDemo: React.FC = () => {
  const costs = [1, 1, 1, 8, 1, 1, 1, 1];
  const step = useLoop(costs.length + 1, 550);
  const total = costs.slice(0, step).reduce((a, b) => a
  const avg = step ? (total / step).toFixed(2) : "—";
  return (
    <Svg caption={`Сделано ${step} операций, суммарно $
      {costs.map((c, i) => (
        <rect key={i} x={14 + i * 30} y={104 - c * 10} w
          fill={i < step ? (c > 2 ? C.bad : C.done) : C
      ))}
      <line x1={8} y1={106} x2={252} y2={106} stroke={C
      <text x={130} y={20} textAnchor="middle" fontSize=
    </Svg>
  );
};

export const NpDemo: React.FC = () => {
  const step = useLoop(2, 1400);
  return (
    <Svg caption={step === 0 ? "Найти решение — долго (
      <ellipse cx={130} cy={64} rx={112} ry={48} fill="
      <ellipse cx={92} cy={64} rx={52} ry={32} fill="#3
      <text x={92} y={64} textAnchor="middle" dominantB
      <text x={206} y={34} textAnchor="middle" fontSize=
```

```
----------------------------------------------------------------
  const step = useLoop(rows * cols + 1, 320);
  return (
    <Svg caption={`Заполняем таблицу подзадач: готово $
      {Array.from({ length: rows }).map((_, r) =>
        Array.from({ length: cols }).map((_, c) => {
          const idx = r * cols + c;
          const filled = idx < step;
          return (
            <g key={idx}>
              <rect x={40 + c * 38} y={22 + r * 32} wid
                fill={idx === step - 1 ? C.active : fil
              {filled && (
                <text x={57 + c * 38} y={36 + r * 32} t
                  fontSize={10} fontWeight="700" fill="
              )}
            </g>
          );
        })
      )}
      <text x={130} y={122} textAnchor="middle" fontSiz
        каждая клетка считается один раз и переиспользу
      </text>
    </Svg>
  );
};


----------------------------------------------------------------
ФАЙЛ 43/74: src/components/hints/MiniDemo.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
----------------------------------------------------------------

import React from "react";
import { ArticulationDemo, BfsDemo, BridgeDemo, DfsDemo
import {
  AmortizedDemo, DsuDemo, HeapDemo, NpDemo, PrefixSumDe
  SparseTableDemo, StackDemo, SuffixLinkDemo, TrieBfsDe
} from "./demosStruct";
```

```
    </div>
  );
};
```

ФАЙЛ 44/74: src/components/hints/SimulatorModal.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО

```
import React, { useEffect } from "react";
import { createPortal } from "react-dom";
import { X } from "lucide-react";
import { getViz } from "../vizRegistry";

interface Props {
  vizId: string | null;
  onClose: () => void;
}

/** Модальное окно с полноценным симулятором — вызывает
export const SimulatorModal: React.FC<Props> = ({ vizId
  useEffect(() => {
    if (!vizId) return;
    const onKey = (e: KeyboardEvent) => {
      if (e.key === "Escape") onClose();
    };
    window.addEventListener("keydown", onKey);
    document.body.style.overflow = "hidden";
    return () => {
      window.removeEventListener("keydown", onKey);
      document.body.style.overflow = "";
    };
  }, [vizId, onClose]);

  const viz = getViz(vizId ?? undefined);
  if (!vizId || !viz) return null;
  const Component = viz.Component;
```

```
import { MiniDemo } from "./MiniDemo";
import { getViz } from "../vizRegistry";

export interface HintAnchor {
  termId: string;
  rect: DOMRect;
}

interface Props {
  anchor: HintAnchor | null;
  onClose: () => void;
  onOpenSimulator: (vizId: string) => void;
  onCardEnter: () => void;
  onCardLeave: () => void;
}

const CARD_W = 320;
const GAP = 10;

export const TermPopover: React.FC<Props> = ({ anchor, 
  const cardRef = useRef<HTMLDivElement>(null);
  const [pos, setPos] = useState<{ top: number; left: n

  useLayoutEffect(() => {
    if (!anchor) {
      setPos(null);
      return;
    }
    const vw = window.innerWidth;
    const vh = window.innerHeight;
    const width = Math.min(CARD_W, vw - 16);
    const height = cardRef.current?.offsetHeight ?? 300

    let left = anchor.rect.left + anchor.rect.width / 2
    left = Math.max(8, Math.min(left, vw - width - 8));

    const below = anchor.rect.top < height + GAP + 8;
    const top = below ? anchor.rect.bottom + GAP : anch
```

```
        <div className="flex items-start gap-2 px-3 pt-
          <h4 className="text-sm font-bold text-indigo-
          <button
            onClick={onClose}
            className="text-slate-500 hover:text-white
            aria-label="Закрыть подсказку"
          >
            <X className="w-4 h-4" />
          </button>
        </div>

        <div className="p-3 space-y-2.5 max-h-[60vh] ov
          <p className="text-[13px] leading-relaxed tex

          {entry.demo && <MiniDemo kind={entry.demo} />

          {entry.formal && (
            <p className="text-[11.5px] leading-relaxed
              {entry.formal}
            </p>
          )}

          {entry.complexity && (
            <div className="text-[11px] font-mono text-
              Сложность: {entry.complexity}
            </div>
          )}

          {viz && simulatorId && (
            <button
              onClick={() => onOpenSimulator(simulatorI
              className="w-full flex items-center justi
            unded-lg py-2 transition-colors"
            >
              <Play className="w-3.5 h-3.5" /> Показать
            </button>
          )}
        </div>
```

```
    };

    const getPotential = (id: string) => {
        if (step >= 2 && step < 7) {
            if (id === 'S') return 0;
            if (id === 'A') return 0;
            if (id === 'B') return -2;
            if (id === 'C') return -1;
        }
        return null;
    };

    const getEdgeColorClass = (e: any) => {
        if (e.u === 'S') {
            if (step === 0 || step === 7) return "opaci
            if (step === 1) return "stroke-amber-500 st
            if (step === 2) return "stroke-pink-500 str
            return "stroke-slate-700 stroke-dasharray-4
        }

        if (e.id === 'AB') {
            if (step === 4) return "stroke-amber-400 st
            if (step > 4) return "stroke-emerald-500";
        }
        if (e.id === 'BC') {
            if (step === 5) return "stroke-amber-400 st
            if (step > 5) return "stroke-emerald-500";
        }
        if (e.id === 'CA') {
            if (step === 6) return "stroke-amber-400 st
            if (step > 6) return "stroke-emerald-500";
        }

        if (step >= 4) return "stroke-slate-600"; // oti
        return "stroke-slate-500";
    };

    const getMarkerUrl = (e: any) => {
```

```
    const isEdgeTextHighlighted = (e: any) => {
        if (e.id === 'AB' && step === 4) return true;
        if (e.id === 'BC' && step === 5) return true;
        if (e.id === 'CA' && step === 6) return true;
        return false;
    };

    const isEdgeTextEmerald = (e: any) => {
        if (e.id === 'AB' && step > 4) return true;
        if (e.id === 'BC' && step > 5) return true;
        if (e.id === 'CA' && step > 6) return true;
        return false;
    };

    const getDesc = () => {
        switch (step) {
            case 0: return (
                <div>
                    <b>Исходный граф.</b> Имеется ребро
                    Если мы запустим быстрый алгоритм Д
                </div>
            );
            case 1: return (
                <div>
                    <b>Шаг 1: Точка отсчета.</b><br/>
                    Добавляем фиктивную вершину S. Соед
            ми</span>.
                    Это наша база для расчета потенциал
                </div>
            );
            case 2: return (
                <div>
                    <b>Шаг 2: Ищем потенциалы $h(v)$.</
                    Запускаем медленный, но мощный алго
            ных вершин:
                    <span className="text-pink-300 ml-1
                </div>
```

```
            default: return "";
        }
    };

    return (
        <div className="bg-slate-800 p-4 rounded-xl bor
            <h4 className="text-sm md:text-base font-bo
                <span className="w-2.5 h-2.5 bg-amber-4
                Визуализация: Перевзвешивание алгоритмо
            </h4>

            <div className="flex flex-col md:flex-row g
                <div className="bg-[#0f172a] flex-1 min
            nter p-4">
                    <svg width="400" height="300" class
                        <defs>
                            <marker id="arrow-slate" vi
            start-reverse">
                                <path d="M 0 0 L 10 5 L
                            </marker>
                            <marker id="arrow-slate-dim
            uto-start-reverse">
                                <path d="M 0 0 L 10 5 L
                            </marker>
                            <marker id="arrow-amber" vi
            start-reverse">
                                <path d="M 0 0 L 10 5 L
                            </marker>
                            <marker id="arrow-pink" vie
            tart-reverse">
                                <path d="M 0 0 L 10 5 L
                            </marker>
                            <marker id="arrow-emerald"
            o-start-reverse">
                                <path d="M 0 0 L 10 5 L
                            </marker>
                        </defs>
```

```
                        {/* Nodes */}
                        {nodes.map(n => {
                            if (!isNodeVisible(n.id)) r
                            const pot = getPotential(n.

                            return (
                                <g key={n.id} className=
                                    <circle
                                        cx={n.x} cy={n.
                                        className={`str
                                        ${n.id === 'S'
                                            ? 'fill-[#2
                                            : 'fill-sla
                                    />
                                    <text x={n.x} y={n.
-white">
                                        {n.label}
                                    </text>

                                    {pot !== null && (
                                        <g transform={`
fade-in zoom-in duration-500">
                                            <rect x="-1
k-500 stroke-1" />
                                            <text x="0"
font-bold fill-pink-400">
                                                {pot}
                                            </text>
                                        </g>
                                    )}
                                </g>
                            )
                        })}
                    </svg>
                </div>

                <div className="w-full md:w-[320px] fle
                    {/* Log Box */}
```

```
----------------------------------------------------------------
}


----------------------------------------------------------------
ФАЙЛ 47/74: src/components/KosarajuViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
----------------------------------------------------------------

import React, { useState, useEffect } from "react";
import {
  Play,
  Pause,
  RotateCcw,
  Layers,
} from "lucide-react";

interface GNode {
  id: number;
  x: number;
  y: number;
  label: string;
}

interface GEdge {
  u: number;
  v: number;
}

const initialNodes: GNode[] = [
  { id: 0, x: 150, y: 100, label: "0" },
  { id: 1, x: 300, y: 100, label: "1" },
  { id: 2, x: 225, y: 220, label: "2" }, // 0->1->2->0
  { id: 3, x: 450, y: 160, label: "3" },
  { id: 4, x: 600, y: 160, label: "4" }, // 3 <-> 4 (Se
];

const initialEdges: GEdge[] = [
  { u: 0, v: 1 },
```

```
      generatedSteps.push({
        phase,
        desc,
        visited: new Set(visited),
        currentNode,
        order: [...order],
        transposed,
        sccMap: { ...sccMap },
        currentSccId,
        activeEdges: transposed
          ? initialEdges.map((e) => ({ u: e.v, v: e.u }
          : [...initialEdges],
      });
    };

    recordStep(
      "init",
      "Инициализация алгоритма Косарайю. Начинаем первы
      null,
      false,
      -1,
    );

    // Adj list
    const adj: number[][] = Array.from({ length: 5 }, (
    initialEdges.forEach((e) => adj[e.u].push(e.v));

    // DFS 1: post-order list
    const dfs1 = (u: number) => {
      visited.add(u);
      recordStep(
        "dfs1",
        `DFS 1: зашли в вершину ${u}, помечаем посещенн
        u,
        false,
        -1,
      );
```

```
    visited.clear();
    let sccId = 0;

    const dfs2 = (u: number, currentScc: number) => {
      visited.add(u);
      sccMap[u] = currentScc;
      recordStep(
        "dfs2",
        `DFS 2: заходим в ${u} на транспонированном гра
        u,
        true,
        currentScc,
      );

      for (const to of adjT[u]) {
        if (!visited.has(to)) {
          dfs2(to, currentScc);
        }
      }
    };

    // Run DFS2 using the order (reversed from back to
    for (let i = order.length - 1; i >= 0; i--) {
      const u = order[i];
      if (!visited.has(u)) {
        sccId++;
        recordStep(
          "dfs2",
          `DFS 2: берем следующую непосещенную из стека
          u,
          true,
          sccId,
        );
        dfs2(u, sccId);
      } else {
        recordStep(
          "dfs2",
          `DFS 2: вершина ${u} из стека уже покрашена в
```

```
      setIsPlaying(false);
    }
    return () => clearInterval(timer);
  }, [isPlaying, currentStepIdx, steps.length]);

  const togglePlay = () => setIsPlaying(!isPlaying);
  const reset = () => {
    setIsPlaying(false);
    setCurrentStepIdx(0);
  };

  const getSccColor = (id: number) => {
    if (id === 1) return "fill-emerald-600 stroke-emera
    if (id === 2) return "fill-indigo-600 stroke-indigo
    return "fill-slate-800 stroke-slate-500";
  };

  return (
    <div className="bg-slate-900 rounded-xl border bord
      <div className="bg-slate-800 p-4 border-b border-
        <h3 className="font-bold text-white flex items-
          <Layers className="w-5 h-5 text-emerald-400" 
          Визуализатор Алгоритма Косарайю (Поиск SCC)
        </h3>

        <div className="flex items-center gap-2 bg-slat
          <button
            onClick={togglePlay}
            className="flex items-center gap-2 px-3 py-
            d-500 text-white"
          >
            {isPlaying ? (
              <Pause className="w-4 h-4 ml-1" />
            ) : (
              <Play className="w-4 h-4 ml-1" />
            )}
            {isPlaying ? "Пауза " : "Старт "}
          </button>
```

```
              </marker>
              <marker
                id="arrow-transposed"
                viewBox="0 0 10 10"
                refX="22"
                refY="5"
                markerWidth="6"
                markerHeight="6"
                orient="auto-start-reverse"
              >
                <path d="M 0 1 L 10 5 L 0 9 z" fill="#6
              </marker>
            </defs>

            {/* Edges */}
            {initialEdges.map((edge, idx) => {
              const u = initialNodes.find((n) => n.id =
              const v = initialNodes.find((n) => n.id =

              // If transposed, draw reversed coordinat
              const x1 = step.transposed ? v.x : u.x;
              const y1 = step.transposed ? v.y : u.y;
              const x2 = step.transposed ? u.x : v.x;
              const y2 = step.transposed ? u.y : v.y;

              const isSccEdge =
                step.sccMap[edge.u] &&
                step.sccMap[edge.v] &&
                step.sccMap[edge.u] === step.sccMap[edg
              let stroke = "#475569";
              let markerId = "arrow";

              if (step.transposed) {
                stroke = isSccEdge ? "#818cf8" : "#4f46
                markerId = "arrow-transposed";
              } else {
                if (
                  step.currentNode === edge.u ||
```

```
                  classes = getSccColor(sccId);
                } else if (isCurrent) {
                  classes = "fill-amber-600 stroke-amber-
                } else if (isVisited && step.phase === "d
                  classes = "fill-emerald-700 stroke-emera
                }

                return (
                  <g key={node.id}>
                    <circle
                      cx={node.x}
                      cy={node.y}
                      r="20"
                      className={`transition-all duration
                      strokeWidth="3"
                    />
                    <text
                      x={node.x}
                      y={node.y}
                      dy=".3em"
                      textAnchor="middle"
                      fill="white"
                      fontSize="13px"
                      fontWeight="bold"
                      className="select-none"
                    >
                      {node.label}
                    </text>

                    {sccId && (
                      <text
                        x={node.x}
                        y={node.y - 28}
                        textAnchor="middle"
                        fill="#10b981"
                        fontSize="10px"
                        fontWeight="bold"
                      >
```

```
                ) : (
                  [...step.order].reverse().map((val, i
                    <div
                      key={idx}
                      className="bg-indigo-950 border b
            ex items-center gap-1"
                    >
                      {val}
                    </div>
                  ))
                )}
              </div>
            </div>

            {/* List of actions log */}
            <div>
              <span className="text-[10px] text-slate-5
                ЛОГ ОПЕРАЦИЙ:
              </span>
              <div className="space-y-1.5 max-h-[150px]
                {steps
                  .slice(0, currentStepIdx + 1)
                  .reverse()
                  .slice(0, 5)
                  .map((s, idx) => (
                    <div
                      key={idx}
                      className={`text-[11px] p-1.5 rou
                    >
                      {s.desc}
                    </div>
                  ))}
              </div>
            </div>
          </div>

          <div className="mt-4 pt-3 border-t border-sla
            <span>
```

```
  x: number;
  y: number;
  color: string; // 'none', 'red', 'blue', 'purple', 'm
}

interface Edge {
  id: string;
  u: number;
  v: number;
  weight: number;
  status: 'pending' | 'inspecting' | 'included' | 'reje
  color: string;
}

interface StepLog {
  title: string;
  description: string;
  type: 'info' | 'success' | 'warning' | 'error';
}

// 9 Vertices layout
const initialVertices: Vertex[] = [
  { id: 0, label: 'A', x: 20, y: 20, color: 'none' },
  { id: 1, label: 'B', x: 50, y: 13, color: 'none' },
  { id: 2, label: 'C', x: 80, y: 20, color: 'none' },
  { id: 3, label: 'D', x: 18, y: 50, color: 'none' },
  { id: 4, label: 'E', x: 50, y: 50, color: 'none' },
  { id: 5, label: 'F', x: 82, y: 50, color: 'none' },
  { id: 6, label: 'G', x: 20, y: 80, color: 'none' },
  { id: 7, label: 'H', x: 50, y: 87, color: 'none' },
  { id: 8, label: 'I', x: 80, y: 80, color: 'none' },
];

// 12 Edges connecting the 9 vertices
const initialEdges: Edge[] = [
  { id: '0-1', u: 0, v: 1, weight: 2, status: 'pending'
  { id: '3-4', u: 3, v: 4, weight: 3, status: 'pending'
  { id: '1-2', u: 1, v: 2, weight: 4, status: 'pending'
```

```
    },
    // Step 2: Include A-B
    () => {
      setVertices(prev => prev.map(v => v.id === 0 || v
      setEdges(prev => prev.map(e => e.id === '0-1' ? {
      setLogs(prev => [{
        title: 'Успех: Ребро A-B в остове',
        description: 'Вершины A and B окрашены в красны
        type: 'success'
      }, ...prev]);
    },
    // Step 3: Inspect D-E (3)
    () => {
      setEdges(prev => prev.map(e => e.id === '3-4' ? {
      setLogs(prev => [{
        title: 'Шаг 2: Берем второе ребро D-E (вес 3)',
        description: 'Оно соединяет две другие бесцветн
        type: 'info'
      }, ...prev]);
    },
    // Step 4: Include D-E
    () => {
      setVertices(prev => prev.map(v => v.id === 3 || v
      setEdges(prev => prev.map(e => e.id === '3-4' ? {
      setLogs(prev => [{
        title: 'Успех: Ребро D-E в остове',
        description: 'Вершины D и E окрашены в синий цв
        type: 'success'
      }, ...prev]);
    },
    // Step 5: Inspect B-C (4)
    () => {
      setEdges(prev => prev.map(e => e.id === '1-2' ? {
      setLogs(prev => [{
        title: 'Шаг 3: Берем ребро B-C (вес 4)',
        description: 'Оно соединяет красную вершину B и
        type: 'info'
      }, ...prev]);
```

```
      }, ...prev]);
    },
    // Step 10: Reject B-E
    () => {
      setEdges(prev => prev.map(e => e.id === '1-4' ? {
      setLogs(prev => [{
        title: 'Отказ: Цикл обнаружен!',
        description: 'Мы выбрасываем ребро B-E. Иначе п
        type: 'error'
      }, ...prev]);
    },
    // Step 11: Inspect E-F (7)
    () => {
      setEdges(prev => prev.map(e => e.id === '4-5' ? {
      setLogs(prev => [{
        title: 'Шаг 6: Берем ребро E-F (вес 7)',
        description: 'Оно соединяет фиолетовую вершину E
        type: 'info'
      }, ...prev]);
    },
    // Step 12: Include E-F
    () => {
      setVertices(prev => prev.map(v => v.id === 5 ? {
      setEdges(prev => prev.map(e => e.id === '4-5' ? {
      setLogs(prev => [{
        title: 'Успех: Ребро E-F в остове',
        description: 'Вершина F окрашена в фиолетовый ц
        type: 'success'
      }, ...prev]);
    },
    // Step 13: Inspect D-G (8)
    () => {
      setEdges(prev => prev.map(e => e.id === '3-6' ? {
      setLogs(prev => [{
        title: 'Шаг 7: Берем ребро D-G (вес 8)',
        description: 'Оно соединяет фиолетовую вершину D
        type: 'info'
      }, ...prev]);
```

```
    // Step 18: Include G-H
    () => {
      setVertices(prev => prev.map(v => v.id === 7 ? {
      setEdges(prev => prev.map(e => e.id === '6-7' ? {
      setLogs(prev => [{
        title: 'Успех: Ребро G-H в остове',
        description: 'Вершина H перекрашена в фиолетовы
        type: 'success'
      }, ...prev]);
    },
    // Step 19: Inspect E-H (11) - Cycle!
    () => {
      setEdges(prev => prev.map(e => e.id === '4-7' ? {
      setLogs(prev => [{
        title: 'Шаг 10: Берем ребро E-H (вес 11)',
        description: 'Оба конца E и H уже фиолетовые. Cl
        type: 'warning'
      }, ...prev]);
    },
    // Step 20: Reject E-H
    () => {
      setEdges(prev => prev.map(e => e.id === '4-7' ? {
      setLogs(prev => [{
        title: 'Отказ: Цикл обнаружен!',
        description: 'Выбрасываем ребро E-H.',
        type: 'error'
      }, ...prev]);
    },
    // Step 21: Inspect H-I (12)
    () => {
      setEdges(prev => prev.map(e => e.id === '7-8' ? {
      setLogs(prev => [{
        title: 'Шаг 11: Берем ребро H-I (вес 12)',
        description: 'Соединяет фиолетовую H и последнюю
        type: 'info'
      }, ...prev]);
    },
    // Step 22: Include H-I
```

```
      setEdges(prev => prev.map(e => e.id === '3-4' ? {
        ..e, status: 'in_heap' } : e));
      setHeapQueue(['A-D (5)', 'B-E (6)', 'E-F (7)', 'D
      setLogs(prev => [{
        title: 'Успех: Плесень захватила вершину D',
        description: 'В кучу добавлены новые пути из D:
        type: 'success'
      }, ...prev]);
    },
    // Step 4: Pop A-D (5)
    () => {
      setEdges(prev => prev.map(e => e.id === '0-3' ? {
      setLogs(prev => [{
        title: 'Шаг 3: Куча выдает ребро A-D (вес 5)',
        description: 'Плесень расширяется в сторону вер
        type: 'info'
      }, ...prev]);
    },
    // Step 5: Include A-D, add A's edges to heap
    () => {
      setVertices(prev => prev.map(v => v.id === 0 ? {
      setEdges(prev => prev.map(e => e.id === '0-3' ? {
        eap' } : e));
      setHeapQueue(['A-B (2)', 'B-E (6)', 'E-F (7)', 'D
      setLogs(prev => [{
        title: 'Успех: Плесень поглотила вершину A',
        description: 'Новое ребро A-B(2) добавлено в ку
        type: 'success'
      }, ...prev]);
    },
    // Step 6: Pop A-B (2)
    () => {
      setEdges(prev => prev.map(e => e.id === '0-1' ? {
      setLogs(prev => [{
        title: 'Шаг 4: Куча выдает ребро A-B (вес 2)',
        description: 'Плесень стремительно бежит к верш
        type: 'info'
      }, ...prev]);
```

```
        title: 'Шаг 6: Куча выдает ребро B-E (вес 6)',
        description: 'ВНИМАНИЕ! Плесень проверяет верши
        type: 'warning'
      }, ...prev]);
    },
    // Step 11: Reject B-E
    () => {
      setEdges(prev => prev.map(e => e.id === '1-4' ? {
      setHeapQueue(['E-F (7)', 'D-G (8)', 'C-F (9)', 'E
      setLogs(prev => [{
        title: 'Отказ: Игнорируем внутреннее ребро',
        description: 'Ребро B-E отброшено. Плесень не р
        type: 'error'
      }, ...prev]);
    },
    // Step 12: Pop E-F (7)
    () => {
      setEdges(prev => prev.map(e => e.id === '4-5' ? {
      setLogs(prev => [{
        title: 'Шаг 7: Куча выдает ребро E-F (вес 7)',
        description: 'Плесень из центра Токио (E) тянет
        type: 'info'
      }, ...prev]);
    },
    // Step 13: Include E-F, add F's edges
    () => {
      setVertices(prev => prev.map(v => v.id === 5 ? {
      setEdges(prev => prev.map(e => e.id === '4-5' ? {
        eap' } : e));
      setHeapQueue(['D-G (8)', 'C-F (9 - цикл)', 'E-H (
      setLogs(prev => [{
        title: 'Успех: Вершина F захвачена',
        description: 'Ребро F-I(13) добавлено в кучу. Р
        type: 'success'
      }, ...prev]);
    },
    // Step 14: Pop D-G (8)
    () => {
```

```
    // Step 18: Pop G-H (10)
    () => {
      setEdges(prev => prev.map(e => e.id === '6-7' ? {
      setLogs(prev => [{
        title: 'Шаг 10: Куча выдает ребро G-H (вес 10)'
        description: 'Плесень ползет к вершине H.',
        type: 'info'
      }, ...prev]);
    },
    // Step 19: Include G-H, add H's edges
    () => {
      setVertices(prev => prev.map(v => v.id === 7 ? {
      setEdges(prev => prev.map(e => e.id === '6-7' ? {
        eap' } : e));
      setHeapQueue(['E-H (11 - цикл)', 'H-I (12)', 'F-I
      setLogs(prev => [{
        title: 'Успех: Вершина H захвачена',
        description: 'В кучу добавлено H-I(12).',
        type: 'success'
      }, ...prev]);
    },
    // Step 20: Pop E-H (11) - Cycle!
    () => {
      setEdges(prev => prev.map(e => e.id === '4-7' ? {
      setLogs(prev => [{
        title: 'Шаг 11: Куча выдает E-H (вес 11)',
        description: 'Обе вершины E и H уже в плесени.'
        type: 'warning'
      }, ...prev]);
    },
    // Step 21: Reject E-H
    () => {
      setEdges(prev => prev.map(e => e.id === '4-7' ? {
      setHeapQueue(['H-I (12)', 'F-I (13)']);
      setLogs(prev => [{
        title: 'Отказ: Игнорируем E-H',
        description: 'Выбрасываем из кучи.',
        type: 'error'
```

```
      // Node 8(I) -> H-I(12)
      setEdges(prev => prev.map(e =>
        ['0-1', '3-4', '1-2', '4-5', '3-6', '6-7', '7-8
      ));
      setLogs(prev => [{
        title: 'Первая пятилетка: Каждая деревня выбира
        description: 'Каждая из 9 деревень одновременно
           A выбирает A-B(2), D выбирает D-E(3), а G вы
          !',
        type: 'info'
      }, ...prev]);
    },
    // Step 2: Concurrently merge into Kolkhozes (Compo
    () => {
      // Concurrently build chosen roads. Merge into:
      // Component 1: {A, B, C} (colored red)
      // Component 2: {D, E, F, G, H, I} (colored gold/
      setVertices(prev => prev.map(v =>
        [0, 1, 2].includes(v.id) ? { ...v, color: 'red'
      ));
      setEdges(prev => prev.map(e => {
        if (['0-1', '1-2'].includes(e.id)) {
          return { ...e, status: 'included', color: 're
        }
        if (['3-4', '4-5', '3-6', '6-7', '7-8'].include
          return { ...e, status: 'included', color: 'bl
        }
        return e;
      }));
      setLogs(prev => [{
        title: 'Слияние: Деревни объединились в колхозы
        description: 'Все выбранные дороги построены ра
          оз {D, E, F, G, H, I}.',
        type: 'success'
      }, ...prev]);
    },
    // Step 3: Five-Year Plan 2 (Inspecting bridge betw
    () => {
```

```
    activeAlgo === 'prim' ? primSteps :
    boruvkaSteps;

  const handleNextStep = () => {
    if (stepIndex < currentSteps.length) {
      currentSteps[stepIndex]();
      setStepIndex(stepIndex + 1);
    }
  };

  const handleReset = (algo: 'kruskal' | 'prim' | 'boru
    setActiveAlgo(algo);
    setVertices(initialVertices);
    setEdges(initialEdges);
    setStepIndex(0);
    setHeapQueue([]);
    if (algo === 'kruskal') {
      setLogs([
        {
          title: 'Введение в раскраску (Краскал)',
          description: 'Представь, что все 9 вершин гра
          type: 'info',
        },
      ]);
    } else if (algo === 'prim') {
      setLogs([
        {
          title: 'Введение в Плесень (Прима)',
          description: 'Логика «Плесени»: выбираем стар
            едь (Heap) и захватываем соседей!',
          type: 'info',
        },
      ]);
    } else {
      setLogs([
        {
          title: 'Введение в Коллективизацию (Борувка)'
          description: 'Логика «Борувки»: Каждая из 9 д
```

```
      <div className="bg-slate-900/90 border border-sla
        <div className="flex flex-col lg:flex-row justi
          <div>
            <div className="flex items-center gap-2 mb-
              <span className="bg-gradient-to-r from-in
            t-bold uppercase tracking-wider">
                Супер-Интерактив 3 в 1
              </span>
              <h3 className="text-2xl font-bold text-wh
            </div>
            <p className="text-slate-400 text-sm">
              Наблюдай за работой Краскала (Раскраска),
            </p>
          </div>

          {/* Algorithm Tabs */}
          <div className="flex flex-wrap items-center g
            <div className="flex items-center bg-slate-
              <button
                onClick={() => handleReset('kruskal')}
                className={
                  "flex items-center gap-1.5 px-3 py-2
             +
                  (activeAlgo === 'kruskal'
                    ? 'bg-indigo-600 text-white shadow-
                    : 'text-slate-400 hover:text-slate-
                }
              >
                <GitGraph className="w-3.5 h-3.5" />
                <span>Краскал (Билет 18)</span>
              </button>
              <button
                onClick={() => handleReset('prim')}
                className={
                  "flex items-center gap-1.5 px-3 py-2
             +
                  (activeAlgo === 'prim'
                    ? 'bg-emerald-600 text-white shadow
```

```
                  : activeAlgo === 'kruskal'
                  ? "bg-indigo-600 hover:bg-indigo-500 t
                  : activeAlgo === 'prim'
                  ? "bg-emerald-600 hover:bg-emerald-500
                  : "bg-rose-600 hover:bg-rose-500 text
                }
              >
                <SkipForward className="w-4 h-4" />
                <span>{stepIndex === 0 ? 'Начать' : stepIs
              </button>
            </div>
          </div>

          <div className="grid grid-cols-1 lg:grid-cols-3
            {/* Graph Canvas */}
            <div className="lg:col-span-2 bg-slate-950 rou
              ative min-h-[480px] overflow-hidden">

              {/* Legend */}
              <div className="absolute top-4 left-4 bg-sla
              -center gap-3 text-xs">
                <div className="flex items-center gap-1.5
                  <div className="w-3 h-3 rounded-full bg
                </div>
                {activeAlgo === 'kruskal' ? (
                  <>
                    <div className="flex items-center gap
                      <div className="w-3 h-3 rounded-full
                    </div>
                    <div className="flex items-center gap
                      <div className="w-3 h-3 rounded-full
                    </div>
                    <div className="flex items-center gap
                      <div className="w-3 h-3 rounded-full
                    </div>
                  </>
                ) : activeAlgo === 'prim' ? (
                  <>
```

```
                return (
                  <g key={edge.id}>
                    <line
                      x1={u.x + "%"}
                      y1={u.y + "%"}
                      x2={v.x + "%"}
                      y2={v.y + "%"}
                      stroke={strokeColor}
                      strokeWidth={isInspecting ? 4.5 :
                      strokeDasharray={isRejected ? '4,
                      className={"transition-all durati
                    />
                    <text
                      x={(u.x + v.x) / 2 + "%"}
                      y={(u.y + v.y) / 2 + "%"}
                      fill={isInspecting ? '#f59e0b' : 
              : '#94a3b8'}
                      fontSize="4.5"
                      fontFamily="monospace"
                      fontWeight="bold"
                      textAnchor="middle"
                      dominantBaseline="middle"
                      className="select-none filter dro
                      dy="-1.5"
                    >
                      {edge.weight}
                    </text>
                  </g>
                );
              })}
            </svg>

            {/* HTML Overlay for Vertices */}
            <div className="absolute inset-0 w-full h-f
              {vertices.map(vertex => (
                <div
                  key={vertex.id}
                  style={{ top: vertex.y + "%", left: v
```

```
                </p>
                <div className="flex flex-wrap gap-1.5"
                  {heapQueue.length === 0 ? (
                    <span className="text-xs text-slate
                  ) : (
                    heapQueue.map((item, idx) => (
                      <span key={idx} className={"px-2 p
            0 shadow-md animate-pulse" : "bg-slate-800 
                        {item}
                      </span>
                    ))
                  )}
                </div>
              </div>
            )}

            <div className="flex-1 p-4 overflow-y-auto 
              {logs.map((log, idx) => (
                <div
                  key={idx}
                  className={
                    "p-4 rounded-xl border transition-a
                    (log.type === 'success'
                      ? 'bg-emerald-950/40 border-emera
                      : log.type === 'warning'
                      ? 'bg-amber-950/40 border-amber-5
                      : log.type === 'error'
                      ? 'bg-rose-950/40 border-rose-500
                      : 'bg-slate-900/60 border-slate-7
                  }
                >
                  <div className="flex items-center gap
                    {log.type === 'success' && <CheckCi
                    {log.type === 'warning' && <HelpCir
                    {log.type === 'error' && <XCircle c
                    {log.type === 'info' && <Play class
                    <h5 className="font-bold text-sm le
                  </div>
```

```
          </div>
        </div>

        {activeCheatTab === 'kruskal' && (
          <div className="grid grid-cols-1 lg:grid-cols
            <div className="lg:col-span-1 space-y-4">
              <div className="bg-slate-950 p-4 rounded-
                <p className="text-slate-400 text-xs fo
                  <Clock className="w-3.5 h-3.5 text-in
                </p>
                <p className="text-2xl font-black text-
                <p className="text-slate-400 text-xs mt
              </div>
              <div className="bg-slate-950 p-4 rounded-
                <p className="text-slate-400 text-xs fo
                  <Award className="w-3.5 h-3.5 text-in
                </p>
                <p className="text-2xl font-black text-
                <p className="text-slate-400 text-xs mt
              </div>
              <div className="bg-slate-950 p-4 rounded-
                <p className="text-slate-400 text-xs fo
            ожении:</p>
                <p className="text-xs text-slate-300 le
            ди и красим вершины в один цвет, бракуя цик
              </div>
            </div>
            <div className="lg:col-span-2">
              <div className="bg-slate-950 p-4 rounded-
                <p className="text-slate-400 text-xs fo
                  <Code className="w-3.5 h-3.5 text-ind
                </p>
                <pre className="text-xs text-emerald-40
            80 flex-1">
{`# 1. Инициализируем DSU
def find(i):
    if parent[i] == i: return i
    parent[i] = find(parent[i])
```

```
              <div className="bg-slate-950 p-4 rounded-
                <p className="text-slate-400 text-xs fo
            ложении:</p>
                <p className="text-xs text-slate-300 le
            ие чистые вершины через самое дешевое досту
              </div>
            </div>
            <div className="lg:col-span-2">
              <div className="bg-slate-950 p-4 rounded-
                <p className="text-slate-400 text-xs fo
                  <Code className="w-3.5 h-3.5 text-eme
                </p>
                <pre className="text-xs text-emerald-40
            80 flex-1">
{`import heapq

def prim(graph, start):
    visited = [False] * len(graph)
    heap = [(0, start, -1)] # (weight, node, parent)
    mst = []

    while heap:
        w, u, prev = heapq.heappop(heap)
        if visited[u]: continue
        visited[u] = True
        if prev != -1: mst.append((prev, u, w))

        for next_w, v in graph[u]:
            if not visited[v]:
                heapq.heappush(heap, (next_w, v, u))
    return mst`}
                </pre>
              </div>
            </div>
          </div>
        )}

        {activeCheatTab === 'boruvka' && (
```

```
---------------------------------------------------------------
            ru, rv = find(u), find(v)
            if ru != rv:
                if cheapest[ru] == -1 or cheapest[ru][2
                if cheapest[rv] == -1 or cheapest[rv][2
        # Объединяем все колхозы по выбранным мостам
        for bridge in cheapest:
            if bridge != -1 and union(bridge[0], bridge
                mst.append(bridge)`}
                </pre>
              </div>
            </div>
          </div>
        )}
      </div>

    </div>
  );
};


---------------------------------------------------------------
ФАЙЛ 49/74: src/components/MnemonicCards.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРОК
---------------------------------------------------------------

import dijkstraImg from '../assets/dijkstra-kind.jpg';
import bellmanImg from '../assets/bellman-ford-truck.jp
import floydImg from '../assets/floyd-network.jpg';

const cards = [
  {
    img: dijkstraImg,
    alt: 'Добрый робот Дейкстра',
    title: '  Добрый (Дейкстра)',
    desc: 'Быстрый, жадный, но работает только с',
    highlight: 'положительными',
    highlightColor: 'text-emerald-400',
    rest: 'весами. Подсунь отрицательное ребро — сломае
```

```
---------------------------------------------------------------
          <div className="h-48 overflow-hidden">
            <img src={c.img} alt={c.alt}
              className="w-full h-full object-cover gro
          </div>
          <div className="p-4">
            <h3 className={`text-lg font-bold mb-1 ${c.
            <p className="text-slate-400 text-sm">
              {c.desc} <span className={`font-bold ${c.
            </p>
            <div className={`mt-3 text-xs font-mono px-
          </div>
        </div>
      )})}
    </div>
  );
}


---------------------------------------------------------------
ФАЙЛ 50/74: src/components/MobileToc.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
---------------------------------------------------------------

import React, { useEffect, useState } from "react";
import { List, X } from "lucide-react";
import { Sidebar } from "./Sidebar";

interface Props {
  selectedChapterId: string;
  setSelectedChapterId: (id: string) => void;
  /** Заголовок текущей темы — показываем в кнопке, что
  currentTitle: string;
  index: number;
  total: number;
}

/**
 * Мобильное содержание: липкая панель снизу-сверху + в
```

```
      {open && (
        <div className="lg:hidden fixed inset-0 z-[90]"
          <div className="absolute inset-0 bg-slate-950
          <div className="absolute inset-y-0 left-0 w-[
            te-[tocIn_.18s_ease-out]">
            <div className="flex items-center justify-b
              <span className="font-bold text-white tex
              <button
                onClick={() => setOpen(false)}
                className="p-2 -mr-2 text-slate-400 hov
                aria-label="Закрыть содержание"
              >
                <X className="w-5 h-5" />
              </button>
            </div>
            <div className="flex-1 min-h-0">
              <Sidebar
                selectedChapterId={selectedChapterId}
                setSelectedChapterId={setSelectedChapte
                onNavigate={() => setOpen(false)}
              />
            </div>
          </div>
        </div>
      )}
    </>
  );
};
```



```
import React from 'react';
import { BookOpen, Cpu, Sparkles, FileDown, FileText }
```

```
          </button>
          <button
            id="tab-simulator-btn"
            onClick={() => setActiveTab('simulator')}
            className={`flex-1 lg:flex-none flex items-
            uration-200 whitespace-nowrap ${
              activeTab === 'simulator'
                ? 'bg-indigo-600 text-white shadow-lg si
                : 'text-slate-400 hover:text-white'
            }`}
          >
            <Cpu className="w-4 h-4" /> Симулятор
          </button>
          <a
            id="download-pdf-btn"
            href="/export/full_code_all_pages.pdf"
            download="full_code_all_pages.pdf"
            target="_blank"
            rel="noreferrer"
            className="flex items-center justify-center
            over:bg-emerald-600/20 border border-emerald
            title="Скачать полный PDF сборник всех стра
          >
            <FileDown className="w-4 h-4" /> Скачать PD
          </a>
          <a
            id="download-txt-btn"
            href="/export/full_code_all_pages.txt"
            download="full_code_all_pages.txt"
            target="_blank"
            rel="noreferrer"
            className="flex items-center justify-center
            :bg-sky-600/20 border border-sky-500/30 tra
            title="Скачать полный TXT файл со всем кодо
          >
            <FileText className="w-4 h-4" /> TXT
          </a>
        </div>
```

```
              function intersect(a:number,b:number,c:nu
                if (a===c||a===d||b===c||b===d) return
                const p1=findPt(a),p2=findPt(b),p3=find
                function ccw(ax:number,ay:number,bx:num
                  return (cy-ay)*(bx-ax)>(by-ay)*(cx-ax
                }
                return ccw(p1.x,p1.y,p3.x,p3.y,p4.x,p4.
                  && ccw(p1.x,p1.y,p2.x,p2.y,p3.x,p3.y)
              }
              const crosses: number[] = [];
              for(let i=0;i<edges.length;i++) {
                for(let j=i+1;j<edges.length;j++) {
                  if (intersect(edges[i][0],edges[i][1]
                    if (!crosses.includes(i)) crosses.p
                    if (!crosses.includes(j)) crosses.p
                  }
                }
              }
              const lines = []; const circles = [];
              for(let i=0;i<edges.length;i++) {
                const p1=findPt(edges[i][0]),p2=findPt(
                const xing = crosses.includes(i);
                lines.push(<line key={`l${i}`} x1={p1.x
                  stroke={xing ? "#f43f5e" : "#4f46e5"}
              }
              const labels = ["A","B","C","D","E"];
              for(const p of pts) {
                circles.push(<g key={`c${p.id}`}>
                  <circle cx={p.x} cy={p.y} r={8} fill=
                  <text x={p.x} y={p.y+3} textAnchor="m
                </g>);
              }
              return <>{lines}{circles}</>;
            })()}
          </svg>
          <div className="absolute bottom-2 left-2 righ
             z-20">
            V=5, E=10 {'>'} 3V-6 = 9 {'-->'} {'\u274C'}
```

```
----------------------------------------------------------------
                const p1=findPt(edges[i][0]),p2=findPt(
                const xing=crosses.includes(i);
                lines.push(<line key={`l${i}`} x1={p1.x
                  stroke={xing?"#f43f5e":"#4f46e5"} str
              }
              const labels = ["\u04141","\u04142","\u04
              for(const p of pts) {
                circles.push(<g key={`c${p.id}`}>
                  <circle cx={p.x} cy={p.y} r={8} fill=
                  <text x={p.x} y={p.y+3} textAnchor="m
                </g>);
              }
              return <>{lines}{circles}</>;
            })()}
          </svg>
          <div className="absolute bottom-2 left-2 righ
             z-20">
            V=6, E=9 {'>'} 2V-4 = 8 (двудольный) {'-->'
          </div>
        </div>
      </div>

      <div className="bg-amber-950/40 border border-amb
        ⚠  Запомни: K5 и K3,3 — два кита непланарности.
        гомеоморфный K5 или K3,3 — он непланарен. Теоре
      </div>
    </div>
  );
}


----------------------------------------------------------------
ФАЙЛ 53/74: src/components/QueueViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
----------------------------------------------------------------

import React, { useState, useCallback } from 'react';
import { Sparkles } from 'lucide-react';
```

```
  const [isServing, setIsServing]          = useState(fa
  const [shakeEmpty, setShake]             = useState(fa
  const [log, setLog]                      = useState<st

  const addLog = (msg: string) => setLog(prev => [msg,

  // ENQUEUE: Встать в конец очереди (хвост / Tail)
  const enqueue = useCallback(() => {
    if (isServing) return;
    if (queue.length >= 6) {
      addLog('⚠ Хвост очереди уже на улице, повару тяже
      setShake(true);
      setTimeout(() => setShake(false), 400);
      return;
    }

    const preset = PRESETS[counter % PRESETS.length];
    counter++;

    const newGuest: Customer = {
      id: Date.now().toString(),
      ...preset,
      animState: 'in',
    };

    setQueue(prev => [...prev, newGuest]);
    addLog(`→ ENQUEUE: ${newGuest.name} ${newGuest.emoj

    setTimeout(() => {
      setQueue(prev => prev.map(c => c.id === newGuest.
    }, 450);
  }, [queue, isServing]);

  // DEQUEUE: Выдать суп первому (голова / Head, FIFO)
  const dequeue = useCallback(() => {
    if (isServing) return;
    if (queue.length === 0) {
      setShake(true);
```

```
    setIsServing(false);
    setLog([' Столовая сброшена. Снова голодная толпа!
    setTimeout(() => {
      setQueue(prev => prev.map(c => ({ ...c, animState
    }, 500);
  };

  return (
    <div className="flex flex-col gap-6">
      {/* МНЕМОНИКА / ПРАВИЛО */}
      <div className="bg-indigo-950/40 border border-in
        <div className="text-3xl"> </div>
        <div>
          <h3 className="font-black text-indigo-400 tex
          <p className="text-xs text-slate-300 mt-1 lea
            В очереди за супом всё честно. Кто первый в
            Новые гости встают строго <b>в конец</b> оч
            Здесь нет обхода и блата — только строгий п
          </p>
        </div>
      </div>

      {/* УПРАВЛЕНИЕ */}
      <div className="flex items-center gap-3 flex-wrap
        <button
          onClick={enqueue}
          disabled={isServing}
          className="flex-1 min-w-[150px] bg-gradient-t
            d:opacity-40 disabled:cursor-not-allowed te
            ive:scale-95 transition-all cursor-pointer
        >
          <span> Встать в очередь (ENQUEUE)</span>
        </button>

        <button
          onClick={dequeue}
          disabled={isServing || queue.length === 0}
          className="flex-1 min-w-[150px] bg-gradient-t
```

```
                <span className="text-3xl block">   <
                <span className="text-[10px] font-bol
          ase font-mono tracking-wider">
                  ПОВАР
                </span>
              </div>
            </div>

            {/* РАЗДЕЛИТЕЛЬНАЯ СТРЕЛКА */}
            <div className="flex flex-col items-cente
              <span className="text-2xl animate-pulse
              <span className="text-[8px] font-mono t
            </div>

            {/* СПИСОК ОЧЕРЕДИ */}
            <div className="flex-1 flex items-end gap
              {queue.length === 0 ? (
                <div className="flex-1 flex flex-col
                  <span className="text-4xl mb-1"> </
                  <p className="text-xs font-bold fon
                  <p className="text-[10px]">Все сыты
                </div>
              ) : (
                queue.map((customer, idx) => {
                  const isHead = idx === 0;
                  const isTail = idx === queue.length
                  return (
                    <div
                      key={customer.id}
                      className={`
                        flex flex-col items-center ga
                        ${customer.animState === 'in'
                        ${customer.animState === 'eat
                        ${customer.animState === 'out
                      `}
                    >
                      {/* Облачко мыслей */}
                      <div className={`
```

```
          {/* Пол кухни */}
          <div className="w-full h-3 bg-gradient-to-r f
        </div>

        {/* ПРАВАЯ КОЛОНКА: СЫТЫЕ И ДОВОЛЬНЫЕ */}
        <div className="md:col-span-3 bg-slate-900 roun
          <div className="absolute top-4 left-4 text-[1
            Сытые гости (Архив FIFO)
          </div>

          <div className="absolute top-4 right-4 flex i
            order-emerald-800/80 text-[9px] font-mono">
            <Sparkles className="w-3.5 h-3.5 animate-pu
            <span>СЫТЫ  </span>
          </div>

          <div className="flex-1 flex flex-col justify-
            {servedHistory.length === 0 ? (
              <div className="flex-1 flex flex-col item
                <span className="text-5xl mb-2"> </span
                <p className="text-xs font-bold font-mo
                <p className="text-[10px] mt-1">Здесь б
              </div>
            ) : (
              <div className="flex flex-col gap-1.5 w-f
                {servedHistory.map((item, idx) => (
                  <div
                    key={idx}
                    className="w-full py-1.5 px-3 round
            0 flex items-center gap-2 text-[11px] text-
                  >
                    <span>{item}</span>
                  </div>
                ))}
              </div>
            )}
          </div>
```

```
  is_terminal: boolean;
  words: string[];
  depth: number;
  x?: number;
  y?: number;
}

export const SalmonAutomatonWidget: React.FC = () => {
  const [words, setWords] = useState<string[]>(['CAR',
  const [newWord, setNewWord] = useState('');
  const [nodes, setNodes] = useState<{ [key: number]: T
  const [simulationStep, setSimulationStep] = useState<
  const [bfsQueue, setBfsQueue] = useState<number[]>([]
  const [currentNode, setCurrentNode] = useState<number
  const [speechText, setSpeechText] = useState<string>(
    'Привет! Я Эхо-Лосось Сеня! Введи слова в словарь и
      буль!'
  );
  const [isTalking, setIsTalking] = useState<boolean>(f
  const [isBlinking, setIsBlinking] = useState<boolean>

  // Регулярное моргание рыбы
  useEffect(() => {
    const blinkInterval = setInterval(() => {
      setIsBlinking(true);
      setTimeout(() => setIsBlinking(false), 200);
    }, 4000);
    return () => clearInterval(blinkInterval);
  }, []);

  // Эффект разговора при смене текста
  useEffect(() => {
    setIsTalking(true);
    const timer = setTimeout(() => setIsTalking(false),
    return () => clearTimeout(timer);
  }, [speechText]);

  // Построение начального бора (Trie) без fail-ссылок
```

```
      const childKeys = Object.values(newNodes[u].child
      newNodes[u].y = 40 + depth * paddingY;

      if (childKeys.length === 0) {
        newNodes[u].x = 40 + currentX * paddingX;
        currentX++;
      } else {
        let leftX = -1;
        let rightX = -1;
        childKeys.forEach(v => {
           layoutDFS(v, depth + 1);
           if (leftX === -1) leftX = newNodes[v].x!;
           rightX = newNodes[v].x!;
        });
        newNodes[u].x = leftX === -1 ? 40 + currentX * p
        if (leftX === -1) currentX++;
      }
    };

    layoutDFS(0, 0);

    setNodes(newNodes);
    setSimulationStep(0);
    setBfsQueue([]);
    setCurrentNode(null);
    setSpeechText(
      `Отлично! Я построил базовое префиксное дерево (Б
        S, чтобы расставить fail-ссылки!`
    );
  };

  useEffect(() => {
    buildInitialTrie(words);
  }, []);

  const handleAddWord = (e: React.FormEvent) => {
    e.preventDefault();
    const cleanWord = newWord.trim().toUpperCase().repla
```

```
  };

  // Один шаг BFS
  const stepBFS = () => {
    if (bfsQueue.length === 0) {
      setSimulationStep(2);
      setCurrentNode(null);
      setSpeechText(
        'Ура! Очередь пуста! Мы успешно построили полны
          ь полюбоваться таблицей переходов!'
      );
      return;
    }

    const r = bfsQueue[0];
    const newQueue = bfsQueue.slice(1);
    const updatedNodes = { ...nodes };
    const rNode = updatedNodes[r];
    const childrenIds = Object.values(rNode.children);

    let logs = `Рассматриваем вершину #${r} (${rNode.ch

    for (const [char, u] of Object.entries(rNode.childr
      newQueue.push(u);
      let v = rNode.fail;
      while (v !== 0 && updatedNodes[v].children[char] =
        v = updatedNodes[v].fail;
      }
      const failTarget = updatedNodes[v].children[char]
      updatedNodes[u].fail = failTarget;

      // Рассчитываем терминальную ссылку (term_link)
      if (updatedNodes[failTarget].is_terminal) {
        updatedNodes[u].term_link = failTarget;
      } else {
        updatedNodes[u].term_link = updatedNodes[failTa
      }
```

```
      </div>

      {/* Верхняя панель: Говорящий лосось и диалоговое
      <div className="grid grid-cols-1 md:grid-cols-3 g
        {/* Аватар Лосося (Сеня) с анимацией моргания и
        <div className="flex flex-col items-center just
          <div className="w-40 h-40 relative flex items
            full border border-blue-500/30 shadow-inner
            {/* SVG Лосося */}
            <svg
              viewBox="0 0 200 200"
              className={`w-36 h-36 transition-transfor
                isTalking ? 'scale-105 drop-shadow-[0_0
              }`}
            >
              {/* Тело лосося */}
              <ellipse cx="100" cy="100" rx="70" ry="45
              <path d="M 40 85 Q 100 55 160 85 Q 100 11

              {/* Хвост */}
              <path
                d="M 35 100 L 5 70 L 15 100 L 5 130 Z"
                fill="#be123c"
                className="origin-[35px_100px] animate-
              />

              {/* Плавники */}
              <path d="M 90 55 L 70 30 L 110 55 Z" fill=
              <path d="M 90 145 L 70 170 L 110 145 Z" f
              <path
                d="M 120 115 L 100 135 L 130 130 Z"
                fill="#f87171"
                className="origin-[120px_115px] animate
              />

              {/* Глаз (с морганием) */}
              <circle cx="145" cy="85" r="12" fill="#ff
              {isBlinking ? (
```

```
            <span>Прямое вещание из аквариума:</span>
          </div>

          <p className="text-slate-200 text-base leadin
            {speechText}
          </p>

          <div className="mt-4 pt-3 border-t border-sla
            <div className="text-xs text-slate-400 flex
              <Info className="w-3.5 h-3.5 text-blue-40
              Статус: {simulationStep === 0 ? 'Бор гото
            оен!'}
            </div>

            <div className="flex gap-2">
              {simulationStep === 0 && (
                <button
                  onClick={startBFS}
                  className="bg-emerald-600 hover:bg-em
            -1 shadow-lg shadow-emerald-900/40 transiti
                >
                  <Play className="w-4 h-4" /> Запустит
                </button>
              )}
              {simulationStep === 1 && (
                <button
                  onClick={stepBFS}
                  className="bg-blue-600 hover:bg-blue-
            dow-lg shadow-blue-900/40 transition-transf
                >
                  <span>Шаг BFS</span>
                  <span className="bg-blue-800 px-1.5 p
                </button>
              )}
              {simulationStep === 2 && (
                <button
                  onClick={() => buildInitialTrie(words
                  className="bg-indigo-600 hover:bg-ind
```

```
              {Object.values(nodes).map(node => {
                if (node.id === 0) return null;
                const parent = nodes[node.parent];
                if (!parent.x || !parent.y || !node.x |

                const isCurrent = currentNode === node.

                return (
                  <g key={`edge-${node.id}`}>
                    <line
                      x1={parent.x}
                      y1={parent.y + 16}
                      x2={node.x}
                      y2={node.y - 16}
                      stroke={isCurrent ? '#60a5fa' : '
                      strokeWidth="2"
                      markerEnd="url(#arrowhead)"
                    />
                    <text
                      x={(parent.x + node.x) / 2 - 10}
                      y={(parent.y + node.y) / 2}
                      fill="#94a3b8"
                      fontSize="12"
                      fontWeight="bold"
                    >
                      {node.char}
                    </text>
                  </g>
                );
              })}

              {/* Draw Fail Links */}
              {simulationStep > 0 && Object.values(node.
                if (node.id === 0 || node.fail === 0) r
            n draw all! But maybe wait, we can just dra
                // Wait, all nodes get fail link. Let's
            tep >= 1 and computed it
                // To avoid too many lines, let's draw
```

```
                    stroke = '#34d399';
                  }

                  return (
                    <g key={`node-${node.id}`}>
                      <circle
                        cx={node.x}
                        cy={node.y}
                        r="16"
                        fill={fill}
                        stroke={stroke}
                        strokeWidth="2"
                      />
                      <text
                        x={node.x}
                        y={node.y + 4}
                        fill="white"
                        fontSize={isRoot ? "10" : "12"}
                        fontWeight="bold"
                        textAnchor="middle"
                      >
                        {isRoot ? 'R' : node.char}
                      </text>
                      {!isRoot && (
                        <text
                          x={node.x + 12}
                          y={node.y - 12}
                          fill="#94a3b8"
                          fontSize="9"
                        >
                          {node.id}
                        </text>
                      )}
                    </g>
                  );
                })}
              </svg>
            );
```

```
            utline-none focus:border-blue-500"
            />
            <button
              type="submit"
              className="bg-slate-700 hover:bg-slate-600
            ition-colors"
            >
              <Plus className="w-3.5 h-3.5" /> Добавить
            </button>
          </form>
        </div>

        {/* Таблица Вершин и суффиксных ссылок */}
        <div className="lg:col-span-2 bg-slate-900/50 p
          <h5 className="text-white font-bold mb-3 flex
            <span className="flex items-center gap-1.5"
              <span>  </span> Таблица переходов и fail-
            </span>
            <span className="text-xs text-slate-400 fon
              Всего вершин: {Object.keys(nodes).length}
            </span>
          </h5>

          <div className="max-h-[220px] overflow-y-auto
            <table className="w-full text-left border-c
              <thead>
                <tr className="border-b border-slate-70
                  <th className="py-2 px-3">ID Вершины<
                  <th className="py-2 px-3">Символ</th>
                  <th className="py-2 px-3">Родитель</t
                  <th className="py-2 px-3">fail-ссылка
                  <th className="py-2 px-3">term_link</
                  <th className="py-2 px-3">Дети</th>
                  <th className="py-2 px-3">Слова (dict
                </tr>
              </thead>
              <tbody className="divide-y divide-slate-8
                {Object.values(nodes).map((node) => {
```

```
                        {node.id === 0 ? (
                          <span className="text-slate-50
                        ) : (
                          <span className="bg-indigo-950
                            #{node.fail}
                          </span>
                        )}
                      </td>
                      <td className="py-2.5 px-3">
                        {node.id === 0 || node.term_lin
                          <span className="text-slate-50
                        ) : (
                          <span className="bg-rose-950 i
          0 transition-colors cursor-help" title={`Скр
                            #{node.term_link}
                          </span>
                        )}
                      </td>
                      <td className="py-2.5 px-3 text-sl
                        {Object.keys(node.children).lenc
                          ? '∅'
                          : Object.entries(node.children
                              .map(([c, id]) => `${c}→#
                              .join(', ')}
                      </td>
                      <td className="py-2.5 px-3">
                        {node.words.length === 0 ? (
                          <span className="text-slate-60
                        ) : (
                          <span className="bg-emerald-95
                            {node.words.join(', ')}
                          </span>
                        )}
                      </td>
                    </tr>
                  );
                })}
              </tbody>
```

```
    tree[v] = (tree[2 * v] ?? 0) + (tree[2 * v + 1] ?? 0
  }
  go(1, 0, n - 1);
  return tree;
}

// ===== STEP GENERATORS =====

// 1. Build steps (recursive top-down)
function genBuildSteps(arr: number[]): Step[] {
  const n = arr.length;
  const steps: Step[] = [];
  let id = 0;
  const tree: Record<number, number> = {};

  const CODE = [
    "build(v, l, r) {",           // 1
    "  if (l === r) {",            // 2
    "    tree[v] = A[l]; return", // 3
    "  }",                         // 4
    "  mid = (l + r) >> 1",        // 5
    "  build(2v, l, mid)",         // 6
    "  build(2v+1, mid+1, r)",     // 7
    "  tree[v] = tree[2v]+tree[2v+1]", // 8
    "}",                           // 9
  ];
  void CODE;

  function go(v: number, l: number, r: number) {
    if (l === r) {
      steps.push({ id: id++, desc: `build(${v}, ${l}, $
        , highlightNodes: [v], arrayHighlight: [l, r] }
      tree[v] = arr[l];
      steps.push({ id: id++, desc: `tree[${v}] = ${arr[
        l, r] });
      return;
    }
    const m = (l + r) >> 1;
```

```
    const right = go(2 * v + 1, m + 1, r, ql, qr);
    const res = left + right;
    steps.push({ id: id++, desc: `Возврат в узел ${v}: 
      rgeChildren: [2 * v, 2 * v + 1], arrayHighlight:
    return res;
  }
  const ans = go(1, 0, n - 1, qL, qR);
  steps.push({ id: id++, desc: `  Запрос sum([${qL}..${
    ghlightNodes: [1], arrayHighlight: [qL, qR], result
  return steps;
}

// 3. Bottom-Up Query steps (iterative on implicit tree
function genQueryBottomUpSteps(arr: number[], qL: numbe
  const n = arr.length;
  // Build iterative tree: leaves at [n..2n-1]
  const tree: Record<number, number> = {};
  for (let i = 0; i < n; i++) tree[n + i] = arr[i];
  for (let i = n - 1; i > 0; --i) tree[i] = (tree[2 * i

  const steps: Step[] = [];
  let id = 0;
  let l = qL + n;
  let r = qR + n + 1; // half-open [l, r)
  let res = 0;

  steps.push({ id: id++, desc: `Старт: l = qL + n = ${q
    `, codeLine: 3, treeState: tree, highlightNodes: [l

  while (l < r) {
    const highlights: number[] = [];
    if (l & 1) {
      res += tree[l] ?? 0;
      steps.push({ id: id++, desc: `l=${l} нечётный → l
        0}. Сумма = ${res}. l++.`, codeLine: 5, treeSta
      highlights.push(l);
      l++;
    }
```

```
  "  tree[v] = tree[2*v] + tree[2*v+1];",
  "}"
];
const QUERY_TD_CODE = [
  "query(v, l, r, ql, qr) {",
  "  if (ql > r || qr < l) return 0;",
  "  // -----",
  "  if (ql <= l && r <= qr) return tree[v];",
  "  // -----",
  "  mid = (l + r) >> 1;",
  "  // спускаемся в обоих детей",
  "  return query(left) + query(right);",
  "}"
];
const QUERY_BU_CODE = [
  "queryBU(ql, qr) {",
  "  res = 0;",
  "  l = ql + n; r = qr + n + 1;",
  "  // -----",
  "  while (l < r) {",
  "    if (l & 1) res += tree[l++];",
  "    // -----",
  "    if (r & 1) res += tree[--r];",
  "    l >>= 1; r >>= 1;",
  "  return res; }"
];

// ===== PLAYER COMPONENT =====
function Player({ steps, color }: { steps: Step[]; colo
  const [idx, setIdx] = useState(0);
  const [playing, setPlaying] = useState(false);
  const [speed, setSpeed] = useState(700);

  useEffect(() => { setIdx(0); setPlaying(false); }, [st

  useEffect(() => {
    if (!playing || idx >= steps.length - 1) { if (idx >
    const t = setTimeout(() => setIdx(i => i + 1), speed
```

```
      <div key={nd.v} className="flex flex-col items-ce
        <div className={`flex flex-col items-center just
          ${cls}`}>
          <span className="text-[7px] opacity-60 font-m
          <span className="text-[9px] font-bold">[{nd.l
          <span className="text-xs font-extrabold font-
        </div>
        {(nd.left || nd.right) && (
          <div className="flex flex-col items-center mt
            <div className="w-px h-2 bg-slate-700" />
            <div className="flex justify-around w-full
              {nd.left && renderNode(nd.left)}
              {nd.right && renderNode(nd.right)}
            </div>
          </div>
        )}
      </div>
    );
  }, [step, clr]);

  return (
    <div className={`rounded-2xl border overflow-hidden
      ? 'border-emerald-500/30' : 'border-amber-500/30'
      {/* Header */}
      <div className={`px-4 py-2.5 flex items-center ju
        -emerald-950/50' : 'bg-amber-950/50'} border-b
        <h4 className="font-bold text-sm text-white fle
        <span className="text-[10px] font-mono text-sla
      </div>

      {/* Array strip */}
      <div className="px-3 py-2 bg-slate-950 border-b b
        {arr.map((v, i) => {
          const inRange = step.arrayHighlight && i >= s
          return (
            <div key={i} className={`flex flex-col item
            go' ? 'bg-indigo-950 border-indigo-500' : c
            er-500'} -translate-y-0.5` : 'bg-slate-900
```

```
        )}
      </div>

      {/* Controls */}
      <div className="px-3 py-2.5 bg-slate-950 border-t
        <div className="flex items-center bg-slate-900 
          <button onClick={() => { setPlaying(false); s
            unded" title="Сброс"><RotateCcw className="
          <button onClick={() => { setPlaying(false); s
            0 hover:text-white hover:bg-slate-800 round
          <button onClick={() => setPlaying(!playing)} 
            white transition-all ${playing ? 'bg-rose-5
            {playing ? <Pause className="w-3 h-3" /> : 
          </button>
          <button onClick={() => { setPlaying(false); s
            .5 text-slate-400 hover:text-white hover:bg
            utton>
        </div>
        <select value={speed} onChange={e => setSpeed(N
          text-[9px] font-bold rounded px-1.5 py-1 outl
          <option value={1200}>Медл.</option>
          <option value={700}>Норма</option>
          <option value={350}>Быстро</option>
          <option value={120}>Турбо</option>
        </select>
      </div>

      {/* Progress bar */}
      <div className="h-1 bg-slate-950"><div className=
        + 1) / total) * 100 : 0}%` }} /></div>
    </div>
  );
}

// ═══════════════════════════════════════════════════════
// MAIN EXPORTED COMPONENT
// ═══════════════════════════════════════════════════════
export function SegmentTreeVisualizer() {
```

```
          {/* Array input */}
          <form onSubmit={handleApply} className="flex-
            <div className="flex items-center justify-b
              <label className="text-[10px] font-bold u
            label>
              <button type="button" onClick={randomize}
            "><RefreshCw className="w-3.5 h-3.5" /></bu
            </div>
            <div className="flex gap-2">
              <input
                value={customInput}
                onChange={e => setCustomInput(e.target.
                className="flex-1 bg-slate-900 border b
            one focus:border-indigo-500"
                placeholder="5, 8, 3, 12, 7, 2"
              />
              <button type="submit" className="bg-indig
            ition-colors">Применить</button>
            </div>
            <div className="flex flex-wrap gap-1.5 pt-1
              {arr.map((v, i) => (
                <span key={i} className="bg-slate-900 b
                  <span className="text-slate-500">A[{i
                </span>
              ))}
              <span className="text-xs text-slate-500 f
            </div>
          </form>

          {/* Query range */}
          <div className="bg-slate-950 p-4 rounded-xl b
            <label className="text-[10px] font-bold upp
            <div className="flex items-center gap-2">
              <div className="flex items-center gap-1">
                <span className="text-xs text-slate-400
                <input
                  type="number"
                  min={0}
```

```
        </button>
        <button onClick={() => setView('theory')} classl
          n-colors ${view === 'theory' ? 'border-blue-50
          te-200'}`}>
          <Info className="w-3.5 h-3.5" /> Почему / Сра
        </button>
      </div>

      {/* ---- TAB CONTENT ---- */}
      {view === 'build' && (
        <SimPanel
          key={`build-${arr.join('-')}`}
          title="Построение дерева (рекурсия)"
          icon=" "
          steps={buildSteps}
          code={BUILD_CODE}
          color="indigo"
          arr={arr}
        />
      )}

      {view === 'queries' && (
        <div className="grid grid-cols-1 xl:grid-cols-2
          <SimPanel
            key={`qtd-${arr.join('-')}-${qL}-${qR}`}
            title={`Запрос sum([${qL}..${qR}]) — Сверху
            icon=" "
            steps={queryTDSteps}
            code={QUERY_TD_CODE}
            color="emerald"
            arr={arr}
          />
          <SimPanel
            key={`qbu-${arr.join('-')}-${qL}-${qR}`}
            title={`Запрос sum([${qL}..${qR}]) — Снизу-
            icon=" "
            steps={queryBUSteps}
            code={QUERY_BU_CODE}
```

```
            наче спускаемся в обоих детей.
              </p>
              <div className="space-y-1.5 text-xs">
                <div className="flex justify-between bo
            span className="text-rose-400 font-mono">O(
                <div className="flex justify-between bo
            ><span className="text-rose-400 font-mono">
                <div className="flex justify-between"><
            d-400 font-bold">✓ легко</span></div>
              </div>
            </div>
            <div className="bg-slate-950 p-5 rounded-xl
              <div className="absolute -top-3 left-4 bg
            rder border-amber-500 shadow-md">
                ↑  Запрос Снизу-вверх (Итерация)
              </div>
              <p className="text-xs text-slate-300 mb-3
                Стартуем с двух указателей (l и r) на л
            ём его левого соседа. Поднимаемся пока l {"
              </p>
              <div className="space-y-1.5 text-xs">
                <div className="flex justify-between bo
            span className="text-emerald-400 font-mono"
                <div className="flex justify-between bo
            ><span className="text-emerald-400 font-mon
                <div className="flex justify-between"><
            00 font-bold">✗ сложно</span></div>
              </div>
            </div>
          </div>

          {/* Verdict */}
          <div className="bg-slate-950 p-5 rounded-xl b
            <h4 className="font-bold text-blue-400 mb-2
            <p className="text-sm text-slate-300 leadin
              <strong className="text-white">Итеративны
            кальности. Но если нужны <strong className=
            ame="text-white">рекурсивный (сверху-вниз)<
```

```
    return Array.from(map.entries());
  }, [query]);

  return (
    <aside className="w-full bg-slate-900/80 lg:bg-tran
      <div className="p-4 pb-3 shrink-0">
        <h3 className="text-xs font-bold text-slate-400
          <Compass className="w-4 h-4 text-indigo-400" 
        </h3>
        <div className="relative">
          <Search className="w-4 h-4 text-slate-500 abs
          <input
            value={query}
            onChange={(e) => setQuery(e.target.value)}
            placeholder="Поиск по темам…"
            className="w-full bg-slate-800/70 border bo
            500 focus:outline-none focus:border-indigo-
          />
          {query && (
            <button
              onClick={() => setQuery("")}
              className="absolute right-2 top-1/2 -tran
              aria-label="Очистить поиск"
            >
              <X className="w-3.5 h-3.5" />
            </button>
          )}
        </div>
      </div>

      <div className="flex-1 overflow-y-auto px-3 pb-4 
        {groups.length === 0 && <p className="text-sm t

        {groups.map(([category, items]) => (
          <div key={category}>
            <div className="text-[10px] font-bold upper
            <div className="space-y-1">
              {items.map((chapter) => {
```

```
---------------------------------------------------------------
        <div className="font-bold text-indigo-400 flex
          <Sparkles className="w-3.5 h-3.5" /> Как поль
        </div>
        <p className="leading-relaxed">
          Подчёркнутые термины в тексте раскрываются по
          симулятора.
        </p>
        <p className="text-slate-400 italic">Стрелки ←
      </div>
    </aside>
  );
};


---------------------------------------------------------------
ФАЙЛ 57/74: src/components/Simulator.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
---------------------------------------------------------------

import React, { useState } from 'react';
import { Layers, AlignLeft, Award, Plus, ArrowRight, Ar

interface Plate {
  id: string;
  name: string;
  icon: string;
}

interface Person {
  id: string;
  name: string;
  icon: string;
}

interface Hypothesis {
  id: string;
  text: string;
  probability: number;
```

```
      { name: 'Тарелка от тако', icon: ' ' },
    ];
    const randomPreset = presets[Math.floor(Math.random
    const newPlate = {
      id: Date.now().toString(),
      name: randomPreset.name,
      icon: randomPreset.icon
    };
    setStack(prev => [...prev, newPlate]);
    setPopMessage(`  Положили поверх стопки: ${newPlate
  };

  const popPlate = () => {
    if (stack.length === 0) {
      setPopMessage("⚠  Стопка пуста! Все тарелки вымыт
      return;
    }
    const topPlate = stack[stack.length - 1];
    setStack(prev => prev.slice(0, prev.length - 1));
    setPopMessage(`  Вымыта верхняя тарелка (LIFO): ${t
  };

  const resetStack = () => {
    setStack([
      { id: '1', name: 'Тарелка из-под пасты', icon: '
      { id: '2', name: 'Тарелка из-под пиццы', icon: '
      { id: '3', name: 'Блюдце от торта', icon: ' ' },
    ]);
    setPopMessage("  Стопка тарелок восстановлена.");
  };

  // Queue handlers
  const addPerson = () => {
    const presets = [
      { name: 'Любитель борща', icon: ' ' },
      { name: 'Курьер доставки', icon: ' ' },
      { name: 'Усталый админ', icon: ' ' },
      { name: 'Турист с рюкзаком', icon: ' ' },
```

```
    };
    setHeap(prev => [...prev, item]);
    setNewCustomText('');
    setHeapMessage(`  Добавлена гипотеза с вероятностью
  };

  const popHeap = () => {
    if (heap.length === 0) {
      setHeapMessage("⚠  Куча пуста! Нет кандидатов для
      return;
    }
    // Find highest probability
    let maxIdx = 0;
    for (let i = 1; i < heap.length; i++) {
      if (heap[i].probability > heap[maxIdx].probabilit
        maxIdx = i;
      }
    }
    const best = heap[maxIdx];
    setHeap(prev => prev.filter((_, idx) => idx !== max
    setHeapMessage(`  Выбран самый "тяжелый" кандидат:
  };

  const resetHeap = () => {
    setHeap([
      { id: '1', text: 'Кандидат: "Привет, я искусствен
      { id: '2', text: 'Кандидат: "Привет, я испек вкус
      { id: '3', text: 'Кандидат: "Привет, я испанский
    ]);
    setHeapMessage("  Гипотезы восстановлены.");
  };

  // Sorted heap for display (Max-Heap property view)
  const sortedHeap = [...heap].sort((a, b) => b.probabil

  return (
    <div className="bg-slate-900/90 border border-slate
      <div className="border-b border-slate-800 pb-6 mb
```

```
        >
          <div className="flex items-center gap-3">
            <AlignLeft className={`w-5 h-5 ${activeTab =
            <div className="text-left">
              <div className="font-bold text-sm text-wh
              <div className="text-xs opacity-80">FIFO
            </div>
          </div>
          <span className="text-xs bg-indigo-500/20 tex
            BFS
          </span>
        </button>

        <button
          onClick={() => setActiveTab('heap')}
          className={`flex items-center justify-between
            activeTab === 'heap'
              ? 'bg-emerald-600/20 border-emerald-500 t
              : 'bg-slate-800/60 border-slate-700/60 te
          }`}
        >
          <div className="flex items-center gap-3">
            <Award className={`w-5 h-5 ${activeTab ===
            <div className="text-left">
              <div className="font-bold text-sm text-wh
              <div className="text-xs opacity-80">Приор
            </div>
          </div>
          <span className="text-xs bg-emerald-500/20 te
            Beam Search
          </span>
        </button>
      </div>

      {/* STACK TAB */}
      {activeTab === 'stack' && (
        <div className="space-y-6">
          <div className="bg-slate-800/80 p-5 rounded-x
```

```
            imate-fade-in">
              <span className="inline-block w-2 h-2 rou
              {popMessage}
            </div>
          )}

          {/* Visualization */}
          <div className="bg-slate-950/60 p-6 rounded-2
            >
            {stack.length === 0 ? (
              <div className="text-center py-12 text-sl
                <div className="text-5xl mb-3">   </div
                <p className="text-base font-bold">Стоп
                <p className="text-xs mt-1">Добавьте гр
              </div>
            ) : (
              <div className="w-full max-w-sm flex flex
                <div className="absolute top-0 text-xs
            full flex items-center gap-1">
                  <span>ВЕРШИНА СТЕКА (TOP)</span>
                  <ArrowDown className="w-3.5 h-3.5 ani
                </div>

                {/* Reversed map so top of stack is vis
                {stack.slice().reverse().map((plate, in
                  const isTop = index === 0;
                  return (
                    <div
                      key={plate.id}
                      className={`w-full flex items-cen
                        isTop
                          ? 'bg-gradient-to-r from-blue
            le-102'
                          : 'bg-slate-900/90 border-sla
                      }`}
                    >
                      <div className="flex items-center
                        <span className="text-2xl">{pla
```

```
                onClick={addPerson}
                className="flex-1 md:flex-none flex ite
              -2.5 rounded-xl text-sm font-bold shadow-lg
              >
                <Plus className="w-4 h-4" /> Встать в о
              </button>
              <button
                onClick={servePerson}
                className="flex-1 md:flex-none flex ite
              er border-indigo-500/40 px-4 py-2.5 rounded
              >
                  Налить суп первому
              </button>
              <button
                onClick={resetQueue}
                title="Сбросить"
                className="p-2.5 bg-slate-800 hover:bg-
              tion-colors cursor-pointer"
              >
                <RotateCcw className="w-5 h-5" />
              </button>
            </div>
          </div>

          {dequeueMessage && (
            <div className="bg-indigo-950/60 border bor
            p-3 animate-fade-in">
              <span className="inline-block w-2 h-2 rou
              {dequeueMessage}
            </div>
          )}

          {/* Visualization */}
          <div className="bg-slate-950/60 p-6 rounded-2
            overflow-x-auto">
            {queue.length === 0 ? (
              <div className="text-center py-12 text-sl
                <div className="text-5xl mb-3">  </div>
```

```
                        isFirst ? 'bg-indigo-500/30 tex
                      }`}>
                        {isFirst ? 'Получает суп' : `В
                      </span>
                    </div>
                  );
                })}

                <div className="flex flex-col items-cen
            -600 min-w-[120px] h-[120px]">
                  <Plus className="w-6 h-6 mb-1" />
                  <span className="text-xs font-mono up
                </div>
              </div>
            )}
            <div className="mt-6 text-xs text-slate-500
              ПЕРВЫМ ПРИШЕЛ → ПЕРВЫМ ПОЛУЧИЛ (FIFO) • И
            </div>
          </div>
        </div>
      )}

      {/* HEAP TAB */}
      {activeTab === 'heap' && (
        <div className="space-y-6">
          <div className="bg-slate-800/80 p-5 rounded-x
            tify-between gap-4">
            <div>
              <h3 className="text-lg font-bold text-whi
                <span>  Приоритетная очередь / Куча</sp
                <span className="text-xs font-mono bg-e
            x-Heap</span>
              </h3>
              <p className="text-slate-400 text-xs mt-1
                Здесь не важно, кто пришел первым. Важн
              </p>
            </div>
            <div className="flex items-center gap-3 w-f
```

```
                min="0.01"
                max="0.99"
                step="0.01"
                value={newCustomProb}
                onChange={e => setNewCustomProb(Number(
                className="w-full accent-emerald-500 cu
              />
            </div>
            <div className="md:col-span-3">
              <button
                type="submit"
                disabled={!newCustomText.trim()}
                className="w-full flex items-center just
            erald-500/40 disabled:opacity-50 disabled:cu
            ointer h-10"
              >
                <Plus className="w-4 h-4" /> Добавить в
              </button>
            </div>
          </form>

          {heapMessage && (
            <div className="bg-emerald-950/60 border bo
             gap-3 animate-fade-in">
              <span className="inline-block w-2 h-2 rou
              {heapMessage}
            </div>
          )}

          {/* Visualization */}
          <div className="bg-slate-950/60 p-6 rounded-2
            >
            {sortedHeap.length === 0 ? (
              <div className="text-center py-12 text-sla
                <div className="text-5xl mb-3"> </div>
                <p className="text-base font-bold">Куча
                <p className="text-xs mt-1">Добавьте ги
              </div>
```

```
                        <div className="flex items-center
                          {/* Custom progress bar */}
                          <div className="hidden sm:block
                            <div
                              className={`h-full rounded-
                              style={{ width: `${percent}%
                            ></div>
                          </div>
                          <span className={`font-mono font
                            isTop ? 'text-emerald-400 text
                          }`}>
                            {percent}%
                          </span>
                        </div>
                      </div>
                    );
                  }))}
                </div>
              )}
              <div className="mt-6 text-xs text-slate-500
                НЕ ВАЖНО, КТО ПРИШЕЛ ПЕРВЫМ • ВАЖЕН САМЫЙ
              </div>
            </div>
          </div>
        )}
      </div>
    );
};
```

ФАЙЛ 58/74: src/components/SplayTreeViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО

```
import React, { useState } from "react";
import {
```

```
  y.left = x;
  return y;
};

// Clone tree helper
const cloneTree = (node: SplayNode | null): SplayNode |
  if (!node) return null;
  return {
    key: node.key,
    left: cloneTree(node.left),
    right: cloneTree(node.right),
    x: node.x,
    y: node.y,
  };
};

export const SplayTreeViz: React.FC = () => {
  // Let's create an elegant initial unbalanced or semi
  // Values: 10, 20, 30, 40, 50, 60
  const createInitialTree = (): SplayNode => {
    let r: SplayNode | null = null;
    const initialKeys = [40, 20, 50, 10, 30, 60];
    initialKeys.forEach((key) => {
      r = insertBST(r, key);
    });
    return r!;
  };

  const [tree, setTree] = useState<SplayNode>(createIni
  const [inputValue, setInputValue] = useState<string>(
  const [log, setLog] = useState<string[]>([
    "Дерево инициализировано. Попробуйте кликнуть на уз
  ]);
  const [highlightedNode, setHighlightedNode] = useStat

  // Splay operation that tracks steps!
  const splayStepByStep = (
    root: SplayNode | null,
```

```
    }

    // Now let's implement standard Splay logic recursi
    const splayAction = (
      node: SplayNode | null,
      k: number,
    ): SplayNode | null => {
      if (!node || node.key === k) return node;

      if (k < node.key) {
        if (!node.left) return node;

        if (k < node.left.key) {
          // Zig-Zig (Left-Left)
          node.left.left = splayAction(node.left.left, 
          node = rightRotate(node);
          steps.push(
            `Вращение Zig-Zig (Правый поворот родителя 
          );
        } else if (k > node.left.key) {
          // Zig-Zag (Left-Right)
          node.left.right = splayAction(node.left.right
          if (node.left.right) {
            node.left = leftRotate(node.left);
            steps.push(`Компонент Zig-Zag: левый поворо
          }
        }

        if (!node.left) return node;
        steps.push(
          `Финальный поворот Zig (Правое вращение корня
        );
        return rightRotate(node);
      } else {
        if (!node.right) return node;

        if (k < node.right.key) {
          // Zag-Zig (Right-Left)
```

```
    const val = parseInt(inputValue);
    if (isNaN(val)) return;

    // Standard BST insert, then splay!
    let newTree = cloneTree(tree);
    newTree = insertBST(newTree, val);
    const { newRoot, steps } = splayStepByStep(newTree,
    if (newRoot) {
      setTree(newRoot);
    }
    setLog([`Вставили вершину [${val}].`, ...steps]);
    setInputValue("");
    setHighlightedNode(val);
  };

  const handleFind = (e: React.FormEvent) => {
    e.preventDefault();
    const val = parseInt(inputValue);
    if (isNaN(val)) return;
    handleSplay(val);
    setInputValue("");
  };

  const resetTree = () => {
    setTree(createInitialTree());
    setHighlightedNode(null);
    setLog(["Дерево возвращено в исходное состояние."])
  };

  // Assign coordinate positions using a simple recursi
  const computeCoordinates = (
    node: SplayNode | null,
    x: number,
    y: number,
    levelWidth: number,
  ): void => {
    if (!node) return;
    node.x = x;
```

```
          stroke="#475569"
          strokeWidth="2"
        />,
      );
      lines = lines.concat(renderLines(node.right));
    }
    return lines;
  };

  const renderNodes = (node: SplayNode | null): React.R
    if (!node) return [];
    let circles: React.ReactNode[] = [];
    const isHighlighted = highlightedNode === node.key;

    circles.push(
      <g
        key={`n-${node.key}`}
        className="cursor-pointer group"
        onClick={() => handleSplay(node.key)}
        data-title={`Кликните, чтобы выполнить Splay(${
      >
        <circle
          cx={node.x}
          cy={node.y}
          r="18"
          fill={isHighlighted ? "#4f46e5" : "#1e293b"}
          stroke={isHighlighted ? "#818cf8" : "#64748b"
          strokeWidth={isHighlighted ? "3" : "2"}
          className="transition-all duration-300 group-
        />
        <text
          x={node.x}
          y={node.y}
          dy=".3em"
          textAnchor="middle"
          fill="white"
          fontSize="12px"
          fontWeight="bold"
```

```
            <form onSubmit={handleFind} className="flex
              <input
                type="number"
                placeholder="Поиск..."
                value={inputValue}
                onChange={(e) => setInputValue(e.target
                className="w-24 bg-slate-950 text-white
            00 focus:outline-none"
              />
              <button
                type="submit"
                onClick={handleFind}
                className="bg-indigo-600 hover:bg-indig
              >
                Поиск / Splay
              </button>
              <button
                type="button"
                onClick={(e) => {
                  e.preventDefault();
                  const val = parseInt(inputValue);
                  if (!isNaN(val)) {
                    let newTree = cloneTree(tree);
                    newTree = insertBST(newTree, val);
                    const { newRoot, steps } = splayStep
                    if (newRoot) setTree(newRoot);
                    setLog([`Вставили [${val}].`, ...st
                    setInputValue("");
                    setHighlightedNode(val);
                  }
                }}
                className="bg-indigo-950 hover:bg-slate
            0"
              >
                Вставить
              </button>
            </form>
```

```
            </p>
            <p>
                <span className="font-semibold text-sla
              колено. Вращаем узел в разные стороны.
            </p>
          </div>
        </div>
      </div>

      {/* Answers Section inside the visualizer widget
      <div className="bg-slate-950/80 p-5 border-t bord
        <h4 className="text-indigo-400 font-bold text-s
          <HelpCircle className="w-4 h-4 text-indigo-40
          Разбор экзаменационных вопросов (Без нытья):
        </h4>

        <div className="grid grid-cols-1 md:grid-cols-3
          <div className="bg-slate-900/60 p-4 rounded-l
            <div>
              <p className="font-bold text-slate-300 mb
                1. Отсутствующий элемент
              </p>
              <p className="text-slate-400 leading-rela
                Допустим, мы ищем{" "}
                <code className="text-indigo-400 font-s
                его нет. Алгоритм спустится к листу, на
                неудачей (например,{" "}
                <code className="text-indigo-400">[20]<
                <code className="text-indigo-400">[30]<
                поднимет в корень{" "}
                <strong className="text-white">
                  последний успешно просмотренный узел
                </strong>
                , на котором он споткнулся! Это оставля
                сверху.
              </p>
            </div>
            <div className="mt-3 text-[10px] text-indig
```

```
---------------------------------------------------------------
                операций. Поэтому амортизированное врем
                <strong className="text-white">O(log n)
              </p>
            </div>
            <div className="mt-3 text-[10px] text-emera
              Каждое "растяжение" компенсируется "склад
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};


---------------------------------------------------------------
ФАЙЛ 59/74: src/components/StackViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
---------------------------------------------------------------

import React, { useState, useCallback } from 'react';
import { ArrowUp, Sparkles } from 'lucide-react';

interface Plate {
  id: string;
  name: string;
  emoji: string;
  color: string;
  isDirty: boolean;
  animState: 'in' | 'idle' | 'washing' | 'clean';
}

const PLATE_PRESETS = [
  { name: 'Тарелка из-под спагетти', emoji: ' ', color:
  { name: 'Тарелка из-под пиццы', emoji: ' ', color: 'f
  { name: 'Блюдце от торта', emoji: ' ', color: 'from-p
  { name: 'Тарелка от тако', emoji: ' ', color: 'from-y
  { name: 'Тарелка из-под суши', emoji: ' ', color: 'fr
```

```
      ...preset,
      isDirty: true,
      animState: 'in',
    };

    setDirtyStack(prev => [...prev, newPlate]);
    addLog(`  PUSH: Положили ${newPlate.emoji} сверху с

    setTimeout(() => {
      setDirtyStack(prev => prev.map(p => p.id === newP
    }, 500);
  }, [dirtyStack, isWashing]);

  // POP: Помыть верхнюю тарелку (LIFO)
  const popPlate = useCallback(() => {
    if (isWashing) return;
    if (dirtyStack.length === 0) {
      setShake(true);
      addLog('⚠ POP: Нечего мыть! Все тарелки чистые.')
      setTimeout(() => setShake(false), 400);
      return;
    }

    setIsWashing(true);
    const topPlate = dirtyStack[dirtyStack.length - 1];

    addLog(`  POP: Начинаем мыть верхнюю тарелку с ${top

    // Step 1: Start washing animation (sponge and soap
    setDirtyStack(prev => prev.map(p => p.id === topPla
    setShowSoap(true);

    setTimeout(() => {
      // Step 2: Wash complete. Plate becomes clean and
      const cleanPlate: Plate = {
        ...topPlate,
        isDirty: false,
        animState: 'clean',
```

```
            Попытаешься вытащить нижнюю — всё рухнет!
          </p>
        </div>
      </div>

      {/* УПРАВЛЕНИЕ */}
      <div className="flex items-center gap-3 flex-wrap
        <button
          onClick={pushPlate}
          disabled={isWashing}
          className="flex-1 min-w-[150px] bg-gradient-t
            acity-40 disabled:cursor-not-allowed text-w
            ale-95 transition-all cursor-pointer flex i
        >
          <span>  Положить грязную (PUSH)</span>
        </button>

        <button
          onClick={popPlate}
          disabled={isWashing || dirtyStack.length === 
          classNamе="flex-1 min-w-[150px] bg-gradient-t
            opacity-40 disabled:cursor-not-allowed text
            ve:scale-95 transition-all cursor-pointer f
        >
          <span>  Помыть верхнюю (POP)</span>
        </button>

        <button
          onClick={reset}
          classNamе="px-5 py-3.5 rounded-2xl bg-slate-8
            r-pointer border border-slate-700"
        >
          Сброс
        </button>
      </div>

      {/* ИНТЕРАКТИВНАЯ КУХНЯ */}
      <div classNamе="grid grid-cols-1 md:grid-cols-12 
```

```
              <div
                className="absolute right-8 flex items-
                style={{ bottom: `${24 + topIndex * 44}
              >
                <span className="text-xs font-mono text
            er gap-1">
                  <ArrowUp className="w-3.5 h-3.5" />
                  <span>ВЕРШИНА (TOP)</span>
                </span>
                <span className="text-blue-400 animate-
              </div>

              {/* Тарелки (отрисовываем в обратном поря
              {dirtyStack.slice().reverse().map((plate,
                const idx = dirtyStack.length - 1 - rev
                const isTop = idx === topIndex;
                return (
                  <div
                    key={plate.id}
                    className={`
                      w-full max-w-[280px] h-10 rounded
            hadow-md select-none transition-all duratio
                      ${plate.color}
                      ${plate.animState === 'in' ? 'ani
                      ${plate.animState === 'washing' ?
                      ${isTop ? 'ring-4 ring-blue-500/5
                    `}
                  >
                    <span className="text-lg">{plate.em
                    <span className="text-slate-800 tex
                    {isTop && (
                      <span className="text-[9px] bg-bl
                        LIFO
                      </span>
                    )}
                  </div>
                );
              })}
```

```
---------------------------------------------------------------
              </div>
            )}
          </div>

          {/* Столешница сушилки */}
          <div className="w-full h-4 bg-gradient-to-r f
        </div>
      </div>

      {/* ЛОГ ДЕЙСТВИЙ */}
      <div className="bg-slate-900 border border-slate-
        <div className="text-[10px] font-mono text-slat
        {log.map((entry, idx) => (
          <div
            key={idx}
            className={`text-xs font-mono flex items-ce
          >
            {idx === 0 ? <span className="text-blue-500
            <span>{entry}</span>
          </div>
        ))}
      </div>
    </div>
  );
};


---------------------------------------------------------------
ФАЙЛ 60/74: src/components/StringAlgorithmsViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
---------------------------------------------------------------

import { useState, useEffect, useRef, useMemo } from "r
import { Play, Pause, SkipForward, Undo, RefreshCw } fr

function generateKmpSteps(pattern: string, text: string
    if (!pattern) pattern = " ";
    const s = pattern + "#" + text;
```

```
    if (!pattern) pattern = " ";
    const s = pattern + "#" + text;
    const n = s.length;
    const z = new Array(n).fill(0);
    const steps: any[] = [];

    let l = 0, r = 0;
    steps.push({ i: 0, l, r, z: [...z], desc: `Начало. 

    for (let i = 1; i < n; i++) {
        steps.push({ i, l, r, z: [...z], desc: `Шаг i=$

        if (i <= r) {
            steps.push({ i, l, r, z: [...z], desc: `Вни
            !` });
            z[i] = Math.min(r - i + 1, z[i - l]);
            steps.push({ i, l, r, z: [...z], desc: `Пре
            }. Копипаста сработала.` });
        }

        steps.push({ i, l, r, zTarget: z[i], zSource: i
        while (i + z[i] < n && s[z[i]] === s[i + z[i]])
            steps.push({ i, l, r, zTarget: z[i], zSourc
            падают! Окно растёт.` });
            z[i]++;
        }

        if (i + z[i] < n) {
            steps.push({ i, l, r, zTarget: z[i], zSourc
            [i]]}' разные. Стоп.` });
        } else {
            steps.push({ i, l, r, z: [...z], desc: `Упёр
        }

        steps.push({ i, l, r, z: [...z], desc: `Фиксиру

        if (i + z[i] - 1 > r) {
            l = i;
```

```
            timer.current = setInterval(() => {
                setStepIdx(prev => {
                    if (prev >= steps.length - 1) { set
                    return prev + 1;
                });
            }, 1200);
        }
        return () => { if (timer.current) clearInterval
    }, [autoPlay, steps.length]);

    const playPause = () => setAutoPlay(!autoPlay);
    const nextStep = () => { setAutoPlay(false); setSte
    const prevStep = () => { setAutoPlay(false); setSte
    const reset = () => { setAutoPlay(false); setStepId

    const step = steps[stepIdx] || steps[0];

    return (
        <div className="space-y-4">
            <div className="flex items-center justify-b
                <div className="flex bg-slate-900 borde
                    <button onClick={() => setMode("kmp
                        className={`px-3 py-1.5 text-xs
            e" : "text-slate-400 hover:text-slate-300"}
                        КМП (π-ФУНКЦИЯ)
                    </button>
                    <button onClick={() => setMode("z")
                        className={`px-3 py-1.5 text-xs
            " : "text-slate-400 hover:text-slate-300"}`
                        Z-ФУНКЦИЯ
                    </button>
                </div>

                <div className="flex items-center gap-2
                    <div className="bg-slate-800 p-1 fl
                        <span className="text-[10px] te
                        <input value={pattern} onChange
            xt-white text-xs font-bold font-mono px-2 p
```

```
-900/40");
                    }
                }

                return (
                    <div key={index} className=
                        {/* L/R markers for Z *
                        {mode === "z" && step.l
                            <div className="abs
                        )}
                        {mode === "z" && step.r
                            <div className="abs
                        )}

                        {/* Index */}
                        <span className={`text-

                        {/* Char Box */}
                        <div className={`w-8 h-
xtColor} font-mono text-lg transition-color
                            {char}
                        </div>

                        {/* Value array box */}
                        <div className="text-[1
slate-700/50 font-mono font-bold">
                            {mode === "kmp" ? s
                        </div>

                        {/* KMP fingers */}
                        {mode === "kmp" && step
                            <div className="abs
 z-20">
                                <span classNam
                            </div>
                        )}
                        {mode === "kmp" && step
                            <div classNam="abs
```

```
                    )
                })}
            </div>
        </div>

        <div className="flex flex-col sm:flex-row ga
            {/* Controls */}
            <div className="flex bg-slate-900 borde
                <button onClick={prevStep} disabled=
        rounded-lg disabled:opacity-30 disabled:hove
                    <Undo strokeWidth={3} size={16}
                </button>
                <button onClick={playPause} classNa
        x items-center justify-center min-w-[80px]"
                    {autoPlay ? <><Pause size={16} -
                </button>
                <button onClick={nextStep} disabled=
        r:bg-slate-800 rounded-lg disabled:opacity-
                    <SkipForward strokeWidth={3} si
                </button>
                <button onClick={reset} className="
        border-slate-800">
                    <RefreshCw strokeWidth={3} size=
                </button>
            </div>

            {/* Description Log */}
            <div className="flex-1 bg-slate-900/50 
        erflow-hidden">
                <div className="absolute top-0 righ
                    <span className="font-mono text
                </div>
                <div className="text-[10px] text-sl
        {steps.length}]</div>
                <div className="text-sm text-slate-
                    {step.desc}
                </div>
            </div>
```

```
---------------------------------------------------------------
            </div>
        </div>
    );
}


---------------------------------------------------------------
ФАЙЛ 61/74: src/components/TopologicalSortViz.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРОК
---------------------------------------------------------------

import React, { useState, useEffect } from "react";
import {
  Play,
  Pause,
  RotateCcw,
  Clock,
} from "lucide-react";

interface TNode {
  id: number;
  x: number;
  y: number;
  label: string;
  name: string;
}

interface TEdge {
  u: number;
  v: number;
}

const dagNodes: TNode[] = [
  { id: 0, x: 100, y: 150, label: "0", name: "Матан 1 (
  { id: 1, x: 280, y: 70, label: "1", name: "Дискретка"
  { id: 2, x: 280, y: 230, label: "2", name: "Линейный
  { id: 3, x: 460, y: 150, label: "3", name: "Алгоритмы
  { id: 4, x: 640, y: 150, label: "4", name: "Машинное
```

```
        dfsStack: [...dfsStack],
      });
    };

    recordStep(
      "Начало топологической сортировки. Используем кла
      null,
    );

    // Adj list
    const adj: number[][] = Array.from({ length: 5 }, (
    dagEdges.forEach((e) => adj[e.u].push(e.v));

    const dfs = (u: number) => {
      visited.add(u);
      dfsStack.push(u);
      recordStep(
        `DFS: зашли в вершину ${u} ("${dagNodes[u].name
        u,
      );

      for (const to of adj[u]) {
        if (!visited.has(to)) {
          dfs(to);
        } else if (!fullyProcessed.has(to)) {
          // Explaining potential cycle detected (not i
          recordStep(
            `⚠  Внимание: Рёбра в серую вершину ${to} o
            u,
          );
        }
      }

      fullyProcessed.add(u);
      dfsStack.pop();
      topoList.push(u); // prepend behavior: we write i
      recordStep(
        `DFS: закончили обработку '${dagNodes[u].name}'
```

```
      }, 1600);
    } else {
      setIsPlaying(false);
    }
    return () => clearInterval(timer);
  }, [isPlaying, currentStepIdx, steps.length]);

  const togglePlay = () => setIsPlaying(!isPlaying);
  const reset = () => {
    setIsPlaying(false);
    setCurrentStepIdx(0);
  };

  const getTopoListString = () => {
    return [...step.topoList]
      .reverse()
      .map((id) => dagNodes[id].name)
      .join(" → ");
  };

  return (
    <div className="bg-slate-900 rounded-xl border bord
      <div className="bg-slate-800 p-4 border-b border-
        <h3 className="font-bold text-white flex items-
          <Clock className="w-5 h-5 text-indigo-400" />
          Топологическая сортировка (Зависимости обучен
        </h3>

        <div className="flex items-center gap-2 bg-slat
          <button
            onClick={togglePlay}
            className="flex items-center gap-2 px-3 py-
            500 text-white"
          >
            {isPlaying ? (
              <Pause className="w-4 h-4 ml-1" />
            ) : (
              <Play className="w-4 h-4 ml-1" />
```

```
              >
                <path d="M 0 1 L 10 5 L 0 9 z" fill="#8
              </marker>
            </defs>

            {/* Edges */}
            {dagEdges.map((edge, idx) => {
              const u = dagNodes.find((n) => n.id === e
              const v = dagNodes.find((n) => n.id === e

              const isActive =
                (step.currentNode === edge.u || step.cu
                step.visited.has(edge.v);

              return (
                <line
                  key={`edge-${idx}`}
                  x1={u.x}
                  y1={u.y}
                  x2={v.x}
                  y2={v.y}
                  stroke={isActive ? "#818cf8" : "#3341
                  strokeWidth={isActive ? "3" : "2"}
                  markerEnd={`url(#${isActive ? "dag-ar
                  className="transition-all duration-30
                />
              );
            })}

            {/* Nodes */}
            {dagNodes.map((node) => {
              const isCurrent = step.currentNode === no
              const isVisited = step.visited.has(node.i
              const isProcessed = step.fullyProcessed.h

              let fill = "#1e293b";
              let stroke = "#334155";
              let textColor = "text-slate-400";
```

```
                  ({node.label}) {node.name.split(" "
                </text>
                <text
                  x={node.x}
                  y={node.y + 12}
                  textAnchor="middle"
                  fill="#94a3b8"
                  fontSize="9px"
                  className="pointer-events-none"
                >
                  {node.name.split(" ").slice(1).join
                </text>
              </g>
            );
          })}
        </svg>
      </div>

      {/* Console column */}
      <div className="lg:col-span-4 bg-slate-950 p-5 
        00 overflow-y-auto">
        <h4 className="text-xs font-bold text-slate-4
          Статус сортировки
        </h4>

        <div className="bg-slate-900 border border-sl
          <p className="text-xs text-indigo-300 min-h
            {step.desc}
          </p>
        </div>

        <div className="flex-1 space-y-4 overflow-y-a
          {/* Topological List output */}
          <div>
            <span className="text-[10px] text-slate-5
              РЕЗУЛЬТАТ (МАТРУТ ОБУЧЕНИЯ):
            </span>
            <div className="bg-slate-900/60 p-2.5 rou
```

```
---------------------------------------------------------------
                onClick={() => setCurrentStepIdx((prev)
                className="bg-slate-900 border border-s
            t-white"
              >
                Назад
              </button>
              <button
                disabled={currentStepIdx >= steps.lengt
                onClick={() => setCurrentStepIdx((prev)
                className="bg-slate-900 border border-s
            t-white"
              >
                Вперед
              </button>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};


---------------------------------------------------------------
ФАЙЛ 62/74: src/components/vizRegistry.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
---------------------------------------------------------------

import React from "react";

import { ArticulationPointsViz } from "./ArticulationPo
import BellmanFordViz from "./BellmanFordViz";
import BfsViz from "./BfsViz";
import { DPVisualizer } from "./DPVisualizer";
import { DfsBridgesSimulator } from "./DfsBridgesSimula
import DijkstraViz from "./DijkstraViz";
import { EulerSimulator } from "./EulerSimulator";
import FloydViz from "./FloydViz";
```

```
  "queue-bfs": {
    title: "Очередь и обход в ширину",
    hint: "Очередь слева, живой BFS по графу справа.",
    Component: () => (
      <div className="space-y-10">
        <QueueViz />
        <BfsViz />
      </div>
    ),
  },
  "heap-beam-search": {
    title: "Куча (priority queue)",
    hint: "Добавляйте гипотезы — куча сама держит лучшу
    Component: HeapViz,
  },
  "segment-trees": {
    title: "Дерево отрезков",
    hint: "Запросы и обновления на отрезке за O(log n).
    Component: SegmentTreeVisualizer,
  },
  "sparse-table": {
    title: "Разреженная таблица (1D и 2D)",
    hint: "Наведите курсор на любую ячейку — подсветитс
    Component: SparseTableViz,
  },
  "prefix-sums-2d": {
    title: "Префиксные суммы (1D и 2D)",
    hint: "Наведите на число — увидите, из чего оно сло
    Component: PrefixSumViz,
  },
  "dynamic-programming": {
    title: "Динамическое программирование",
    hint: "Таблица подзадач заполняется на глазах.",
    Component: DPVisualizer,
  },
  "splay-tree": { title: "Splay-дерево", Component: Spl
  "graph-dfs-bfs": { title: "DFS и BFS на графе", Compo
  "top-sort": { title: "Топологическая сортировка", Com
```

```
----------------------------------------------------------------
  "string-z-func": { title: "Z-функция", Component: () =
  "aho-corasick": {
    title: "Ахо-Корасик",
    hint: "Бор, суффиксные ссылки и BFS-обход по уровня
    Component: () => (
      <div className="space-y-10">
        <WaterfallAnimationWidget />
        <SalmonAutomatonWidget />
      </div>
    ),
  },
  intro: { title: "Мнемокарточки", Component: MnemonicC
};

export const getViz = (id?: string): VizEntry | undefin


----------------------------------------------------------------
ФАЙЛ 63/74: src/components/WaterfallAnimationWidget.tsx
КАТЕГОРИЯ: Интерактивные компоненты и симуляторы | СТРО
----------------------------------------------------------------

import React, { useState } from 'react';
import { ArrowRight, CheckCircle2, AlertCircle, Lightbu

// Структура заранее подготовленного красивого бора для
// Словарь: ["HE", "SHE", "HIS", "HERS"]
interface WNode {
  id: number;
  label: string; // путь от корня
  char: string;
  x: number; // Координаты для SVG водопада
  y: number;
  depth: number;
  fail: number;
  term_link: number;
  is_terminal: boolean;
  words: string[];
```

```
  const [textToScan, setTextToScan] = useState<string>(
  const [currentIndex, setCurrentIndex] = useState<numb
  const [currentNode, setCurrentNode] = useState<number
  const [isSearchDone, setIsSearchDone] = useState<bool
  const [searchLog, setSearchLog] = useState<string>(
    'Режим поиска: Введите текст и жмите «Следующий шаг
  );
  const [jumpPath, setJumpPath] = useState<{ from: numb
  const [foundMatches, setFoundMatches] = useState<{ in
  const [activeSearchCodeLine, setActiveSearchCodeLine]

  // === СОСТОЯНИЯ РЕЖИМА ОБУЧЕНИЯ (TRAINING) ===
  // Шаги обучения от 0 до 8 (по числу вершин в боре, и
  const [trainStep, setTrainStep] = useState<number>(0)

  const handleTextChange = (e: React.ChangeEvent<HTMLIn
    const clean = e.target.value.toUpperCase().replace(
    setTextToScan(clean);
    resetSearchSimulation();
  };

  const resetSearchSimulation = () => {
    setCurrentIndex(0);
    setCurrentNode(0);
    setIsSearchDone(false);
    setSearchLog('Симуляция сброшена. Лосось вернулся н
    setJumpPath(null);
    setFoundMatches([]);
    setActiveSearchCodeLine(1);
  };

  const stepSearchSimulation = () => {
    if (currentIndex >= textToScan.length) {
      setIsSearchDone(true);
      setSearchLog('Мы проплыли весь текст! Лосось дово
      setActiveSearchCodeLine(4);
      return;
    }
```

```
    let temp = curr;
    while (temp !== 0) {
        if (WATERFALL_NODES[temp].is_terminal) {
            WATERFALL_NODES[temp].words.forEach((word) =
                currentMatches.push({ index: currentInde
                matchedWordsForLog.push(word);
            });
        }
        temp = WATERFALL_NODES[temp].term_link;
    }

    if (matchedWordsForLog.length > 0) {
      logMsg += `  БИНГО! Найдено слово: ${matchedWords
      setActiveSearchCodeLine(4);
    }

    setCurrentNode(curr);
    setCurrentIndex(currentIndex + 1);
    setSearchLog(logMsg);
    if (currentMatches.length > 0) {
      setFoundMatches((prev) => [...prev, ...currentMat
    }
  };

  // ПОЛНАЯ ПОШАГОВАЯ ИНФОРМАЦИЯ О ПОСТРОЕНИИ ДЕРЕВА (В
  const TRAINING_STEPS_INFO = [
    {
      targetNodeId: 0,
      title: 'Шаг 0 (Depth 0): Вершина #0 (ROOT)',
      desc: 'Начало алгоритма BFS. Корень (ROOT) не обра
        рние вершины сразу инициализируются с fail = RO
      codeLine: 1,
      scoutPos: 0,
      inspectedParentFail: 0,
      checkTargetChar: 'ø',
      checkingBranchesFrom: null,
    },
```

```
      codeLine: 3,
      scoutPos: 0,
      inspectedParentFail: 0,
      checkTargetChar: 'I',
      checkingBranchesFrom: 0,
    },
    {
      targetNodeId: 8,
      title: 'Шаг 5 (Depth 2): Вершина #8 (SH) — КАК СТ
      desc: '  ВАЖНЫЙ ВОПРОС! Обрабатываем #8 (SH) на D
        ). Рыба плывет в ROOT и ищет ветку "H". У корня
         суффикс "H"!',
      codeLine: 4,
      scoutPos: 0,
      inspectedParentFail: 0,
      checkTargetChar: 'H',
      checkingBranchesFrom: 0,
    },
    {
      targetNodeId: 3,
      title: 'Шаг 6 (Depth 3): Вершина #3 (HER)',
      desc: 'Переходим на Depth 3 к #3 (HER). Родитель
        е подходят. fail[HER] = ROOT.',
      codeLine: 3,
      scoutPos: 0,
      inspectedParentFail: 0,
      checkTargetChar: 'R',
      checkingBranchesFrom: 0,
    },
    {
      targetNodeId: 6,
      title: 'Шаг 7 (Depth 3): Вершина #6 (HIS)',
      desc: 'Обрабатываем #6 (HIS) на Depth 3. Родитель
        но ветка "S" → #7 (S) совпадает! fail[HIS] = #7
      codeLine: 4,
      scoutPos: 0,
      inspectedParentFail: 0,
      checkTargetChar: 'S',
```

```
            Анимация водопада: Полное обучение автома
          </h4>
          <p className="text-sm text-slate-400">
            Визуализация пошагового построения всех fa
          </p>
        </div>
      </div>

      {/* Переключатель режимов */}
      <div className="bg-slate-900 p-1.5 rounded-xl bo
        <button
          onClick={() => setActiveMode('training')}
          className={`flex items-center space-x-1.5 p
            activeMode === 'training'
              ? 'bg-indigo-600 text-white shadow-lg sh
              : 'text-slate-400 hover:text-white'
          }`}
        >
          <GitBranch className="w-4 h-4" />
          <span>Режим 1: Полное обучение (BFS от корн
        </button>
        <button
          onClick={() => setActiveMode('search')}
          className={`flex items-center space-x-1.5 p
            activeMode === 'search'
              ? 'bg-blue-600 text-white shadow-lg shad
              : 'text-slate-400 hover:text-white'
          }`}
        >
          <Play className="w-4 h-4" />
          <span>Режим 2: Поиск в тексте</span>
        </button>
      </div>
    </div>

    {/* ПАНЕЛЬ УПРАВЛЕНИЯ В ЗАВИСИМОСТИ ОТ РЕЖИМА */}
    {activeMode === 'training' ? (
      /* ПАНЕЛЬ РЕЖИМА ОБУЧЕНИЯ */
```

```
              </button>
              <button
                onClick={() => setTrainStep((prev) => M
                disabled={trainStep === TRAINING_STEPS_
                className="flex-1 bg-indigo-600 hover:b
            ition-all shadow-lg shadow-indigo-900/50"
              >
                {trainStep === TRAINING_STEPS_INFO.leng
              </button>
            </div>
          </div>

          {/* Блок с кодом построения fail-ссылки и опи
          <div className="lg:col-span-2 bg-slate-900/90
            >
            <div>
              <h5 className="text-white font-bold mb-3
                <span className="flex items-center gap-
                  <span>⚡</span> Код BFS построения fai
                </span>
                <span className="text-xs bg-indigo-950
                  Вершина: #{targetTrainingNode.id} ({t
                </span>
              </h5>
              <div className="bg-slate-950 p-4 rounded-
                <div className={`p-1.5 rounded flex ite
            -l-4 border-indigo-500 text-indigo-300 font
                  <span>1. let u = trie[r][char]; // Te
                  {currentTrainInfo.codeLine === 1 && <
             тут</span>}
                </div>
                <div className={`p-1.5 rounded flex ite
            -l-4 border-indigo-400 text-indigo-200 font
                  <span>2. let v = fail[r]; // Подъем к
            ainInfo.inspectedParentFail].label})</span>
                  {currentTrainInfo.codeLine === 2 && <
            к родителю</span>}
                </div>
```

```
              <h5 className="text-white font-bold mb-2 t
                <span>✍ </span> Текст для сканирования
              </h5>
              <p className="text-xs text-slate-400 mb-4
                Введи текст, чтобы лосось просканировал
              </p>
              <input
                type="text"
                value={textToScan}
                onChange={handleTextChange}
                maxLength={15}
                placeholder="ВВЕДИТЕ ТЕКСТ..."
                className="w-full bg-slate-800 border bo
            cus:outline-none focus:border-blue-500"
              />
            </div>

            <div>
              <div classNameе="mb-4 flex flex-wrap gap-1
                <span classNameе="text-xs text-slate-400
                {textToScan.split('').map((char, idx) =>
                  <span
                    key={idx}
                    classNameе={`w-7 h-8 flex items-cent
                      idx === currentIndex
                        ? 'bg-blue-600 text-white shado
                        : idx < currentIndex
                        ? 'bg-emerald-950/60 text-emera
                        : 'bg-slate-800 text-slate-500'
                    }`}
                  >
                    {char}
                  </span>
                ))}
                {currentIndex >= textToScan.length && (
                  <span classNameе="bg-emerald-600 text-
                    ГОТОВО ✓
                  </span>
```

```
            border-blue-400 text-blue-200 font-bold' :
                 <span>3. curr = trie[curr].get(char)
                 {activeSearchCodeLine === 3 && <span
          ем по течению</span>}
              </div>
              <div className={`p-1.5 rounded flex ite
          l-4 border-emerald-500 text-emerald-300 fon
                 <span>4. if (is_terminal[curr]) repor
                 {activeSearchCodeLine === 4 && <span
          овладение!</span>}
              </div>
            </div>
          </div>

          <div>
            <h5 className="text-slate-300 font-bold m
              <AlertCircle className="w-3.5 h-3.5 tex
            </h5>
            <div className="bg-slate-800 p-3.5 rounde
          tems-center">
              {searchLog}
            </div>
          </div>
        </div>
      </div>
    )}

    {/* SVG Анимация водопада */}
    <div className="bg-gradient-to-b from-blue-950 vi
      -hidden shadow-2xl">
      {/* Водопадные фоновые эффекты */}
      <div className="absolute inset-0 opacity-10 bg-
        o-900 to-transparent pointer-events-none"></d
      <div className="absolute top-0 left-1/2 -transl
        <div className="w-8 bg-blue-400 h-full animat
        <div className="w-12 bg-blue-300 h-full anima
        <div className="w-6 bg-blue-500 h-full animat
      </div>
```

```
                <rect x="25" y={level.y - 12} width="14
              />
                <text x="32" y={level.y + 4} fill="#94a
                  {level.label}
                </text>
              </g>
            ))}

            {/* Рендер прямых переходов (потоков воды)
            {Object.entries(TRIE_TRANSITIONS).map(([fro
              const fromNode = WATERFALL_NODES[Number(f
              return Object.entries(children).map(([cha
                const toNode = WATERFALL_NODES[toId];

                // Подсветка в режиме поиска
                let isHighlighted = activeMode === 'sea
            ype === 'normal';

                // Подсветка в режиме обучения (когда и
                let isTrainingExamined = activeMode ===
                let isMatchBranch = isTrainingExamined

                return (
                  <g key={`edge-${fromNode.id}-${toNode
                    {/* Линия течения воды */}
                    <line
                      x1={fromNode.x}
                      y1={fromNode.y}
                      x2={toNode.x}
                      y2={toNode.y}
                      stroke={isHighlighted ? '#3b82f6'
                      strokeWidth={isHighlighted || isT
                      className={isHighlighted || isTra
                    />
                    {/* Символ перехода */}
                    <circle cx={(fromNode.x + toNode.x)
            ined ? (isMatchBranch ? '#10b981' : '#f43f5
                    <text
```

```
              let isJustEstablished = activeMode === 't
            -ссылку

              // Вычисляем изогнутую кривую для красоты
              const dx = targetNode.x - node.x;
              const dy = targetNode.y - node.y;
              const cx = node.x + dx / 2 + (node.id % 2
              const cy = node.y + dy / 2 - 30;

              return (
                <g key={`fail-${node.id}`}>
                  <path
                    d={`M ${node.x} ${node.y} Q ${cx} $
                    fill="none"
                    stroke={isHighlighted ? '#f43f5e' :
                    strokeWidth={isHighlighted || isJus
                    strokeDasharray="6,6"
                    className={isHighlighted || isJustE
                    opacity={isHighlighted || isJustEst
                  />
                  {isJustEstablished && (
                    <text x={cx} y={cy - 10} fill="#10b
            >
                      fail[{node.label}] = {targetNode.
                    </text>
                  )}
                </g>
              );
            })}

            {/* Рендер вершин (бассейнов водопада) */}
            {Object.values(WATERFALL_NODES).map((node) =
              const isCurrent = activeFishPosNode.id ===
              const isTargetChild = activeMode === 'tra
              const hasWords = node.words.length > 0;

              return (
                <g key={`node-${node.id}`} className="c
```

```
                    </g>
                  )}

                  {/* Индикатор словарных слов */}
                  {hasWords && (
                    <g transform={`translate(${node.x +
                      <circle cx="0" cy="0" r="9" fill=
                      <text x="0" y="3" fontSize="10" f
                        *
                      </text>
                    </g>
                  )}
                </g>
              );
            })}

            {/* ОДИН ПЛАВНЫЙ АНИМИРОВАННЫЙ ЭХО-ЛОСОСЬ *
            <g>
              {/* Круг-подложка под рыбу */}
              <circle
                cx={activeFishPosNode.x + 15}
                cy={activeFishPosNode.y - 25}
                r="18"
                fill={activeMode === 'training' ? '#8b5
                stroke="#ffffff"
                strokeWidth="2"
                style={{ transition: 'cx 0.8s cubic-bez
                className="shadow-lg"
              />
              {/* Сама рыба */}
              <text
                x={activeFishPosNode.x + 15}
                y={activeFishPosNode.y - 19}
                fontSize="18"
                textAnchor="middle"
                style={{ transition: 'x 0.8s cubic-bezi
                className="animate-float select-none"
              >
```

```
  );
};
```

РАЗДЕЛ: ВИЗУАЛИЗАТОРЫ (ВЛОЖЕННЫЕ)

ФАЙЛ 64/74: src/components/visualizers/PrefixSumViz.tsx
КАТЕГОРИЯ: Визуализаторы (вложенные) | СТРОК: 330 | РАЗ

```
import React, { useMemo, useState } from "react";

/**
 * Префиксные суммы: 1D и 2D.
 *
 * Главная фишка — наведение на число:
 *  • в 1D наводим на P[i] — подсвечивается кусок масси
 *  • в 1D наводим на a[i] — видно, в какие префиксы он
 *  • в 2D наводим на ячейку — подсвечивается прямоугол
 *    а при выбранном запросе показывается формула вклю
 */

const A1 = [3, 1, 4, 1, 5, 9, 2, 6];

const A2 = [
  [1, 2, 3, 4],
  [5, 6, 7, 8],
  [9, 1, 2, 3],
  [4, 5, 6, 7],
];

const cellBase =
  "flex items-center justify-center rounded-md font-mon
  "w-[clamp(30px,9vw,46px)] h-[clamp(30px,9vw,46px)] te
```

```
  const covered =
    hover?.kind === "pref"
      ? { from: 0, to: hover.i - 1 }
      : hover?.kind === "a"
        ? { from: hover.i, to: hover.i }
        : null;

  const sum = pref[range.r + 1] - pref[range.l];

  return (
    <div className="space-y-5">
      <div className="overflow-x-auto pb-1">
        <div className="inline-block min-w-full">
          {/* индексы */}
          <div className="flex gap-1.5 mb-1 pl-[52px]">
            {A1.map((_, i) => (
              <div key={i} className={`${cellBase} !h-5
                {i}
              </div>
            ))}
          </div>

          {/* массив a */}
          <div className="flex gap-1.5 items-center mb-
            <div className="w-[46px] shrink-0 text-righ
            {A1.map((v, i) => {
              const inCover = covered && i >= covered.f
              const inRange = i >= range.l && i <= rang
              return (
                <div
                  key={i}
                  onMouseEnter={() => setHover({ kind:
                  onMouseLeave={() => setHover(null)}
                  onClick={() => setRange((r) => (i < r
                  className={`${cellBase} cursor-pointe
                    inCover
                      ? "bg-indigo-500 text-white ring-
                      : inRange
```

```
          </div>
        </div>
      </div>

      {/* Пояснение под курсором */}
      <div className="min-h-[62px] bg-slate-900 border i
        {hover?.kind === "pref" ? (
          hover.i === 0 ? (
            <p className="text-slate-300">
              <b className="text-emerald-400">P[0] = 0<
              отрезка, начинающегося с нуля.
            </p>
          ) : (
            <p className="text-slate-300">
              <b className="text-emerald-400">
                P[{hover.i}] = {pref[hover.i]}
              </b>{" "}
              — это сумма всего, что набежало от начала
              <span className="font-mono text-indigo-300
                a[0..{hover.i - 1}] = {A1.slice(0, hove
              </span>
            </p>
          )
        ) : hover?.kind === "a" ? (
          <p className="text-slate-300">
            <b className="text-indigo-400">
              a[{hover.i}] = {A1[hover.i]}
            </b>{" "}
            входит во все префиксы, начиная с{" "}
            <span className="font-mono text-emerald-300
            подвинуть границу запроса.
          </p>
        ) : (
          <p className="text-slate-500">
            Наведите курсор на любое число: у <b classNa
            у <b className="text-indigo-400">a</b> — ку
          </p>
        )}
```

```
  const D = P[rect.r1][rect.c1];
  const B = P[rect.r1][rect.c2 + 1];
  const Cc = P[rect.r2 + 1][rect.c1];
  const Aa = P[rect.r2 + 1][rect.c2 + 1];
  const total = Aa - B - Cc + D;

  const cornerOf = (i: number, j: number) => {
    if (i === rect.r2 + 1 && j === rect.c2 + 1) return
    if (i === rect.r1 && j === rect.c2 + 1) return "B";
    if (i === rect.r2 + 1 && j === rect.c1) return "C";
    if (i === rect.r1 && j === rect.c1) return "D";
    return null;
  };

  return (
    <div className="space-y-5">
      <div className="grid gap-5 lg:grid-cols-2">
        {/* Исходная матрица */}
        <div className="overflow-x-auto">
          <div className="text-xs text-slate-400 mb-2">
          <div className="inline-block">
            {A2.map((row, i) => (
              <div key={i} className="flex gap-1.5 mb-1
                {row.map((v, j) => {
                  const inRect = i >= rect.r1 && i <= r
                  return (
                    <div
                      key={j}
                      onClick={() =>
                        setRect((r) =>
                          i < r.r1 || j < r.c1 ? { r1:
                        )
                      }
                      className={`${cellBase} cursor-po
                        inRect ? "bg-indigo-500 text-wh
                      }`}
                    >
```

```
            ht">
                                {corner}
                              </span>
                            )}
                          </div>
                        );
                      })}
                    </div>
                  )})}
                </div>
              </div>
            </div>

            <div className="min-h-[54px] bg-slate-900 border b
              {hover ? (
                <p className="text-slate-300">
                  <b className="text-amber-400">
                    P[{hover.i}][{hover.j}] = {P[hover.i][hove
                  </b>{" "}
                  — сумма всего прямоугольника от левого верхн
                </p>
              ) : (
                <p className="text-slate-500">Наведите курсор
              )}
            </div>

            <div className="bg-slate-900 border border-indigo
              <p className="font-mono text-sm sm:text-base te
                сумма = <span className="text-emerald-400">A<
                <span className="text-rose-400">C</span> + <sp
                <span className="text-indigo-300 font-bold">{
              </p>
              <p className="text-[11px] text-slate-500 mt-1">
                Вычли два «хвоста» и вернули дважды вычтенный
              </p>
            </div>
          </div>
        );
```

```
  [25, 41, 16, 92,  9, 17, 56, 31],
  [59, 18, 77, 24, 61, 82, 35, 12],
  [73,  8, 38, 85, 47, 95, 19, 58],
  [39, 81, 62, 28, 51, 42, 85, 14]
];

const ST2D: number[][][][] = Array.from({length: 8}, ()
  Array.from({length: 8}, () =>
    Array.from({length: 4}, () =>
      Array(4).fill(0)
    )
  )
);

for (let r = 0; r < 8; r++) {
  for (let c = 0; c < 8; c++) {
    ST2D[r][c][0][0] = val2D[r][c];
  }
}

for (let kx = 0; kx <= 3; kx++) {
  for (let ky = 0; ky <= 3; ky++) {
    if (kx === 0 && ky === 0) continue;
    for (let r = 0; r + (1 << kx) <= 8; r++) {
      for (let c = 0; c + (1 << ky) <= 8; c++) {
         if (kx > 0) {
            ST2D[r][c][kx][ky] = Math.min(
               ST2D[r][c][kx-1][ky],
               ST2D[r + (1 << (kx-1))][c][kx-1][ky]
            );
         } else {
            ST2D[r][c][kx][ky] = Math.min(
               ST2D[r][c][kx][ky-1],
               ST2D[r][c + (1 << (ky-1))][kx][ky-1]
            );
         }
      }
    }
```

```
          <Viz2DQuery key="2d_query" />
        )}
      </AnimatePresence>
    </div>
  );
};

const Viz1D: React.FC = () => {
  const [step, setStep] = useState(0);
  const [hover, setHover] = useState<{ i: number; j: nu

  // Total steps: we animate computing each element for
  const computeSteps: { i: number; j: number }[] = [];
  for (let j = 1; j < LOG; j++) {
    for (let i = 0; i + (1 << j) <= N; i++) {
      computeSteps.push({ i, j });
    }
  }

  const maxStep = computeSteps.length;
  const covered = hover ? { from: hover.i, to: hover.i +

  return (
    <div className="flex flex-col gap-5">
      <div className="flex items-center justify-between
        <div>
          <h3 className="text-lg font-bold text-indigo-
            Построение Sparse Table (Min)
          </h3>
          <p className="text-sm text-slate-400">
            Формула: ST[i][j] = min(ST[i][j-1], ST[i + 1
          </p>
        </div>
        <div className="flex gap-2">
          <button
            onClick={() => setStep(Math.max(0, step - 1
            className="p-2 bg-slate-800 hover:bg-slate-
            disabled={step === 0}
```

```
        </div>
        <div className="mt-2 text-xs min-h-[36px]">
          {hover && covered ? (
            <p className="text-slate-300">
              <b className="text-sky-400">
                ST[{hover.i}][{hover.j}] = {st1D[hover.
              </b>{" "}
              — это минимум на отрезке{" "}
              <span className="font-mono text-sky-300">
                [{covered.from}, {covered.to}]
              </span>{" "}
              длины 2^{hover.j} = {1 << hover.j}:{" "}
              <span className="font-mono text-slate-400
                min({arr1D.slice(covered.from, covered.
              </span>
            </p>
          ) : (
            <p className="text-slate-500">
              Наведите курсор (или тапните) на любое чи
            </p>
          )}
        </div>
      </div>

      <div className="overflow-x-auto pb-2">
        <div className="inline-block min-w-full bg-slat
          <div className="grid grid-cols-[auto_repeat(4
            {/* Headers */}
            <div className="text-slate-500 font-mono te
              i \ j
            </div>
            <div className="text-center font-bold text-
            rap">
              2^0 (L=1)
            </div>
            <div className="text-center font-bold text-
            rap">
              2^1 (L=2)
```

```
                    )
                      isSrc2 = true; // wait, no. Src2 
                    // Let's re-eval exact source:
                    // We are asking if THIS cell [i][j
                    if (currentStep.j - 1 === j) {
                      if (currentStep.i === i) isSrc1 =
                      if (currentStep.i + (1 << (curren
                        isSrc2 = true;
                    }
                  }

                  return (
                    <div key={j} className="relative fl
                      {!isValid ? (
                        <div className="w-[clamp(30px,9
                      ) : (
                        <motion.div
                          onMouseEnter={() => isVisible
                          onMouseLeave={() => setHover(
                          onClick={() => isVisible && s
                          initial={{ opacity: 0, scale:
                          animate={{
                            opacity: isVisible ? 1 : 0,
                            scale: isVisible ? 1 : 0.8,
                          }}
                          className={`w-[clamp(30px,9vw
            clamp(11px,3vw,17px)] font-bold transition-
                            hover && hover.i === i && h
                              ? "bg-sky-500 text-white
                              : isActive
                              ? "bg-indigo-500 text-whi
                              : isSrc1
                                ? "bg-emerald-500 text-
                                : isSrc2
                                  ? "bg-amber-500 text-
                                  : isVisible
                                    ? "bg-slate-800 tex
                                    : "bg-transparent t
```

```
      <div className="bg-slate-900 border border-slate-
        {step > 0 && step <= maxStep ? (
          (() => {
            const c = computeSteps[step - 1];
            const half = 1 << (c.j - 1);
            return (
              <p className="text-lg text-slate-300">
                <span className="text-white font-bold">
                  ST[{c.i}][{c.j}]
                </span>{" "}
                = min(
                <span className="text-emerald-400 font-
                  ST[{c.i}][{c.j - 1}]
                </span>
                ,
                <span className="text-amber-400 font-bo
                  ST[{c.i + half}][{c.j - 1}]
                </span>
                ) &rarr; min(
                <span className="text-emerald-400">{st1
                <span className="text-amber-400">
                  {st1D[c.i + half][c.j - 1]}
                </span>
                ) ={" "}
                <span className="text-indigo-400 font-b
                  {st1D[c.i][c.j]}
                </span>
              </p>
            );
          })()
        ) : (
          <p className="text-slate-500 italic">
            Нажмите далее, чтобы увидеть, как блоки сво
          </p>
        )}
      </div>
    </div>
  );
```

```
              const stepSize = 8 - stepL + 1;
              const isCurr = step === k;

              return (
                <React.Fragment key={step}>
                  {idx > 0 && <div className="text-slate
                  <div className={`flex flex-col items-c
                    <h4 className={`font-bold mb-3 flex 
            }>
                      <span>Матрица блоков {stepL}x{step
                      <span className={`text-[10px] font
            rder-indigo-800/40 text-indigo-300' : 'bg-s
                    </h4>
                    <div className="max-w-full overflow-
                      <div className={`grid gap-[2px] p-
            go-900/40 shadow-[0_0_25px_rgba(99,102,241,
            eColumns: `repeat(${stepSize}, max-content)
                      {Array.from({length: stepSize * st
                         const r = Math.floor(idx / step
                         const c = idx % stepSize;

                         let highlightClass = isCurr ? '
            /50 cursor-pointer' : 'bg-slate-800/50 text
                         let zIndex = 'z-0';

                         const isActive = !!activeCell &
            ll.c + L) && ((r - activeCell.r) % (1 << st

                         if (isActive) {
                           if (isCurr) {
                             highlightClass = `bg-indigo
            {locked && locked.r === r && locked.c === c
                             zIndex = 'z-10';
                           } else {
                             const prevL = 1 << (k - 1);
                             const isTop = r < activeCel
                             const isLeft = c < activeCe
```

```
                </div>
              </React.Fragment>
            )
          }))}
        </div>
      </div>

      <div className="flex-1 min-w-0 flex flex-col gap-
        <div className="bg-slate-900 p-4 sm:p-6 rounded
          <h3 className="text-lg font-bold text-slate-3
          {k === 0 ? (
            <div className="text-slate-400 text-sm lea
              <p className="mb-4">При <span className=
            ы имеют размер <span className="font-bold t
              <div className="bg-[#0a0f1e] rounded-xl
                <span className="text-slate-500">// Ба
                <span className="text-indigo-400">ST</
              </div>
              <p className="mt-6 italic text-indigo-40
                <Play size={14} className="fill-indigo
              </p>
            </div>
          ) : (
            <div className="flex flex-col gap-4 animat
              <div className="bg-[#0a0f1e] rounded-xl
            ate-300 shadow-inner overflow-x-auto">
                <span className="text-slate-500">// Te
                <span className="text-purple-400">int<
            ="text-amber-300">1</span>);<br/><br/>
                <span className="text-slate-500">// Ка
                <span className="text-indigo-400">ST</
                  <span className="text-rose
            e="text-amber-300">1</span>],          <sp
                  <span className="text-blue
            ext-amber-300">1</span>][k-<span className=
            pan><br/>
                  <span className="text-emera
            ="text-amber-300">1</span>][k-<span classNa
```

```
                                <div className="flex ite
                                   <span className="flex
            -500 shadow-[0_0_8px_#10b981]"></div> ST[{a
                                   <span className="text
           >{ST2D[activeCell.r + prevL][activeCell.c][
                                </div>
                                <div className="flex ite
                                   <span className="flex
             shadow-[0_0_8px_#f59e0b]"></div> ST[{activ
                                   <span className="text
           [activeCell.r + prevL][activeCell.c + prevL
                                </div>
                              </>
                           )
                        })()}
                    </div>

                    <div className="pt-3 mt-3 border-t
                       ) = <span className="text-white
           0">{ST2D[activeCell.r][activeCell.c][k][k]}
                    </div>
                    <p className="mt-4 text-[12px] font
           список матриц (слева) вниз, чтобы увидеть,
                 </div>
               ) : (
                 <div className="bg-slate-950/40 border
             mt-2 flex items-center justify-center anim
                      Наведите курсор на матрицу {L}x{L
                 </div>
               )}
             </div>
          )}
        </div>
      </div>
    </div>
  );
};
```

```
                    }))}

                     <div
                       className="absolute border-[3px
          ow-[0_0_15px_rgba(244,63,94,0.3)] border-da
                       style={{
                         top: `${(r1 / 8) * 100}%`,
                         left: `${(c1 / 8) * 100}%`,
                         marginTop: '8px',
                         marginLeft: '8px',
                         height: 'calc(' + (H/8*100) +
                         width: 'calc(' + (W/8*100) +
                       }}
                    />

                    {/* Показываем 4 угла в самой матр
                    <div className="absolute pointer-e
           shadow-[0_0_10px_#f43f5e]" style={{ top: `
          lc(${(Ly / 8) * 100}% - 16px)`, height: `ca
                    <div className="absolute pointer-e
           shadow-[0_0_10px_#3b82f6]" style={{ top: `
           width: `calc(${(Ly / 8) * 100}% - 16px)`,
                    <div className="absolute pointer-e
          ld-400 shadow-[0_0_10px_#10b981]" style={{
          8px)`, width: `calc(${(Ly / 8) * 100}% - 16
                    <div className="absolute pointer-e
          00 shadow-[0_0_10px_#f59e0b]" style={{ top:
          00}% + 8px)`, width: `calc(${(Ly / 8) * 100
                 </div>

                 <div className="bg-slate-950 p-4 roun
          er-slate-800 shadow-[inset_0_2px_10px_rgba(
                      <label className="flex items-cen
                          <span className="text-slate-
                          <div className="flex gap-2">
                            <input type="number" min={
          14 bg-slate-900 border border-slate-700 py-
                            <input type="number" min={
```

```
            "text-slate-500">// Выс: {Lx}</span><br/>
                <span className="text-purple-400">int<
            "text-slate-500">// Шир: {Ly}</span><br/><b

                <span className="text-slate-500">// 2.
                <span className="text-indigo-400">int<
                  <span className="text-slat
            В запроса)</span><br/>
                  <span className="text-rose
            r1</span>][<span className="text-white bg-r

                  <span className="text-slat
            во, чтобы правый край уперся в c2)</span><b
                  <span className="text-blue
            r1</span>][<span className="text-white bg-b

                  <span className="text-slat
            px, чтобы нижний край уперся в r2)</span><b
                  <span className="text-emer
            nded">r2 - Lx + 1</span>][<span className="

                  <span className="text-slat
            и влево от c2)</span><br/>
                  <span className="text-ambe
            ">r2 - Lx + 1</span>][<span className="text
                {'}'});
              </div>

              <div className="w-full flex justify-cent
                <div className="flex flex-col items-ce
                 <h4 className="text-slate-400 font-bo
                    Откуда берутся эти 4 числа: матриц
            border-slate-700 shadow-sm">ST[][][{kx}][{k
                 </h4>
                 <div className="grid gap-[2px] p-2.5
            Columns: `repeat(${matCols}, 1fr)`}}>
                    {Array.from({length: matRows * ma
                        const r = Math.floor(idx / ma
```

```
----------------------------------------------------------------
                        <span className="text-blue-400 m
            n>
                        <span className="text-emerald-40
            span>
                        <span className="text-amber-400 
                     <span className="text-slate-4
                    <span className="ml-3 text-emerald-
            + 1][c1][kx][ky], ST2D[r2 - Lx + 1][c2 - Ly
                </div>
              </div>

            </div>
        </div>
    )
}


================================================================
РАЗДЕЛ: HTML-РАЗМЕТКА (LAYOUT)
================================================================


----------------------------------------------------------------
ФАЙЛ 66/74: src/layout/footer.html
КАТЕГОРИЯ: HTML-разметка (layout) | СТРОК: 137 | РАЗМЕР
----------------------------------------------------------------



            </div>
        </main>
    </div>

    <script>
        function setMode(mode) {
            document.getElementById('btn-mode-normal').
                ? 'px-3 py-1 rounded text-sm font-bold 
                : 'px-3 py-1 rounded text-sm font-bold 

            document.getElementById('btn-mode-wtf').cla
```

```
            if (theme === 'light') document.getElementB
            if (theme === 'terminal') document.getEleme
            );

            document.getElementById('btn-theme-dark').c
            t-white transition-colors' : 'px-3 py-1 rou
            n-colors';
            document.getElementById('btn-theme-light').
            ext-white transition-colors' : 'px-3 py-1 r
            ion-colors';
            document.getElementById('btn-theme-notebook
            -600 text-white transition-colors' : 'px-3 
            ransition-colors';
        }

        // Initialize theme and mode
        setTheme('dark');

        // Setup intersection observer for TOC highligh
        document.addEventListener('DOMContentLoaded', (
            const mobileMenuBtn = document.getElementBy
            const mobileDrawer = document.getElementBy
            const closeDrawerBtn = document.getElementB
            const drawerOverlay = document.getElementBy

            function toggleDrawer() {
                const isClosed = mobileDrawer.classList
                if (isClosed) {
                    mobileDrawer.classList.remove("-tra
                    if (drawerOverlay) {
                        drawerOverlay.classList.remove(
                        // Timeout allows display:block
                        setTimeout(() => drawerOverlay.
                    }
                    document.body.style.overflow = "hid
                } else {
                    mobileDrawer.classList.add("-transl
                    if (drawerOverlay) {
```

```
----------------------------------------------------------------
                                if (link.getAttribute('href
                                    link.classList.add('bg-
                                } else {
                                    link.classList.remove('
                                }
                            });
                        }
                    });
                }, observerOptions);

                document.querySelectorAll('section[id^="que
                    observer.observe(section);
                });
            });
        </script>
</body>
</html>


----------------------------------------------------------------
ФАЙЛ 67/74: src/layout/header.html
КАТЕГОРИЯ: HTML-разметка (layout) | СТРОК: 1169 | РАЗМЕ
----------------------------------------------------------------

<!DOCTYPE html>
<html lang="ru" class="scroll-smooth">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    <title>Пособие по алгебре: От пиццы к предикатам</t
    <script src="https://cdn.tailwindcss.com"></script>
    <script src="https://polyfill.io/v3/polyfill.min.js
    <script id="MathJax-script" async src="https://cdn.
    <script>
        window.MathJax = {
            tex: {
                inlineMath: [['$', '$'], ['\\(', '\\)']
                displayMath: [['$$', '$$'], ['\\[', '\\
```

```
        }
        mjx-container[display="true"] {
            overflow-x: auto;
            overflow-y: hidden;
            max-width: 100%;
            scrollbar-width: none;
            -ms-overflow-style: none;
            -webkit-overflow-scrolling: touch;
        }
        mjx-container[display="true"]::-webkit-scrollba
            display: none;
        }

        @media (max-width: 640px) {
            html { font-size: 13px; }

            .uno-card { width: 40px; height: 60px; font
            .uno-card::before, .uno-card::after { font-
            .uno-card::before { top: 1px; left: 2px; }
            .uno-card::after { bottom: 1px; right: 2px;

            .uno-card.mini { width: 28px !important; he
            us: 4px; }
            .uno-card.mini::before, .uno-card.mini::aft

            .uno-equation .text-2xl { font-size: 1.2rem
            .uno-equation.gap-4 { gap: 0.5rem !importan

            /* Allow horizontal scroll for equations in
            .formula-scroll { flex-wrap: nowrap !import
            px; }
        }
        .uno-inner {
            position: absolute; width: 110%; height: 75
            border-radius: 50%; transform: rotate(-30de
            box-shadow: inset 0 0 5px rgba(0,0,0,0.3);
        }
        .uno-val { position: relative; z-index: 2; }
```

```
            font-family: "Comic Sans MS", "Caveat", "Se
        }
        body.theme-notebook .bg-slate-900, body.theme-n
          late-700 {
            background-color: rgba(255, 255, 255, 0.7)
            border-color: #cbd5e1 !important;
            box-shadow: 2px 2px 5px rgba(0,0,0,0.05);
        }
        body.theme-notebook .text-slate-200, body.theme
          ant; }
        body.theme-notebook .text-slate-400 { color: #5
        body.theme-notebook .border-slate-600, body.the
          : #94a3b8 !important; }
        body.theme-notebook .font-mono { font-family: "
        body.theme-notebook #top-controls { background-


        /* CREEPY BLINKING */
        @keyframes creepy-blink {
            0%, 95%, 98%, 100% { transform: scaleY(1);
            96.5% { transform: scaleY(0.1); }
        }
        .creepy-eye {
            transform-origin: center;
            transform-box: fill-box;
            animation: creepy-blink 4s infinite;
        }
        .creepy-eye-alt {
            transform-origin: center;
            transform-box: fill-box;
            animation: creepy-blink 5.2s infinite;
            animation-delay: 1.5s;
        }
        .creepy-emoji {
            display: inline-block;
            transform-origin: center;
            animation: creepy-blink 3.8s infinite;
        }
```

```
            <path d="M0,0 L6,3 L0,6 Z" fill="#f472b
        </marker>
        <marker id="arrow-cyan" markerWidth="6" mar
            <path d="M0,0 L6,3 L0,6 Z" fill="#2dd4b
        </marker>
        <marker id="arrow-indigo" markerWidth="6" m
            <path d="M0,0 L6,3 L0,6 Z" fill="#818cf
        </marker>
        <marker id="arrow-lime" markerWidth="6" mar
            <path d="M0,0 L6,3 L0,6 Z" fill="#a3e63
        </marker>
        <marker id="arrow-green" markerWidth="6" ma
            <path d="M0,0 L6,3 L0,6 Z" fill="#4ade8
        </marker>

        <!-- ЛЕГО БЛОКИ -->
        <g id="lego-yellow">
            <rect x="0" y="0" width="40" height="20
            <rect x="6" y="-5" width="8" height="6"
            <rect x="26" y="-5" width="8" height="6
            <line x1="2" y1="2" x2="38" y2="2" stro
        </g>
        <g id="lego-red">
            <rect x="0" y="0" width="40" height="20
            <rect x="6" y="-5" width="8" height="6"
            <rect x="26" y="-5" width="8" height="6
            <line x1="2" y1="2" x2="38" y2="2" stro
        </g>
        <g id="lego-rem">
            <rect x="0" y="0" width="40" height="10
            <rect x="6" y="-5" width="8" height="6"
            <rect x="26" y="-5" width="8" height="6
            <line x1="2" y1="2" x2="38" y2="2" stro
        </g>
        <!-- МЯСОРУБКА -->
        <g id="meat-grinder">
            <path d="M 10 40 L 15 20 Q 25 15 35 20
            <rect x="5" y="40" width="40" height="2
```

```
            <div class="h-full flex flex-col p-6 overfl
            lg:rounded-2xl lg:border lg:border-slate-70
                <div class="flex justify-between items-
                    <h3 class="font-black text-transpar
            ider text-sm">
                        Содержание
                    </h3>
                    <button id="close-drawer-btn" class
                        <svg xmlns="http://www.w3.org/2
                            <path stroke-linecap="round
                        </svg>
                    </button>
                </div>
                <nav class="space-y-3 text-sm font-medi

                    <details class="group" open>
                        <summary class="list-none curso
                            <a href="#question-1" class
            er-blue-500 pl-3 font-bold text-blue-300">
                        1. Алгебраические структуры
                    </a>
                        </summary>
                        <div class="pl-6 space-y-2 text
                        <a href="#section-1-1" class="b
                        <a href="#section-1-2" class="b
                        <a href="#section-1-3" class="b
                        <a href="#section-1-4" class="b
                    </div>
                    </details>

                    <details class="group">
                        <summary class="list-none curso
                            <a href="#question-2" class
            er-indigo-500 pl-3 font-bold text-indigo-30
                        2. Комплексные числа
                    </a>
                        </summary>
                        <div class="pl-6 space-y-2 text
```

```
                        <a href="#question-5" class=
er-fuchsia-500 pl-3 font-bold text-fuchsia-
                        5. Группа корней из единицы
                    </a>
                        </summary>
                        <div class="pl-6 space-y-2 text
                        <a href="#q5-def" class="block
                        <a href="#q5-group" class="bloc
                        <a href="#q5-prim" class="block
                        <a href="#q5-crit" class="block
                        <a href="#q5-cyclic" class="blo
                    </div>
                    </details>


                    <details class="group">
                        <summary class="list-none curso
                        <a href="#question-6" class=
er-lime-500 pl-3 font-bold text-lime-300">
                        6. Делимость, НОД и Евклид
                    </a>
                        </summary>
                        <div class="pl-6 space-y-2 text
                        <a href="#q6-1" class="block ho
                        <a href="#q6-2" class="block ho
                        <a href="#q6-3" class="block ho
                        <a href="#q6-4" class="block ho
                        <a href="#q6-5" class="block ho
                        <a href="#q6-6" class="block ho
                    </div>
                    </details>

                    <details class="group">
                        <summary class="list-none curso
                        <a href="#question-7" class=
er-pink-500 pl-3 font-bold text-pink-300">
                            7. Взаимно простые числа
                        </a>
```

```
                        <a href="#q9-3" class="bloc
                        <a href="#q9-4" class="bloc
                    </div>
                </details>

                <details class="group">
                    <summary class="list-none curso
                        <a href="#question-10" clas
            der-rose-500 pl-3 font-bold text-rose-400">
                            10. Делимость и Схема Г
                        </a>
                    </summary>
                    <div class="pl-6 space-y-2 text
                        <a href="#q10-1" class="blo
                        <a href="#q10-2" class="blo
                        <a href="#q10-3" class="blo
            a>
                        <a href="#q10-4" class="blo
                        <a href="#q10-5" class="blo
                    </div>
                </details>

                <details class="group">
                    <summary class="list-none curso
                        <a href="#question-11" clas
            der-cyan-500 pl-3 font-bold text-cyan-400">
                            11. НОД и НОК многочлен
                        </a>
                    </summary>
                    <div class="pl-6 space-y-2 text
                        <a href="#q11-1" class="blo
                        <a href="#q11-2" class="blo
            нственности НОД</a>
                        <a href="#q11-3" class="blo
            ПИРАЛЬ)</a>
                        <a href="#q11-4" class="blo
            вращается!)</a>
                    </div>
```

```
                    <div class="pl-6 mt-2 space-y-2
                        <a href="#q14-1" class="blo
                        <a href="#q14-2" class="blo
                        <a href="#q14-3" class="blo
 deg)</a>
                        <a href="#q14-4" class="blo
        ition-colors border border-rose-500/30">  K
                    </div>
                </details>

                <details class="group">
                    <summary class="list-none curso
                        <a href="#question-15" clas
        der-cyan-500 pl-3 font-bold text-cyan-400">
                            15. Производная. Критер
                        </a>
                    </summary>
                    <div class="pl-6 mt-2 space-y-2
                        <a href="#q15-1" class="blo
                        <a href="#q15-2" class="blo
                        <a href="#q15-3" class="blo
                        <a href="#q15-4" class="blo
                    </div>
                </details>

                <details class="group">
                    <summary class="list-none curso
                        <a href="#question-16" clas
        der-red-500 pl-3 font-bold text-red-400">
                            16. Неприводимые многоч
                        </a>
                    </summary>
                    <div class="pl-6 mt-2 space-y-2
                        <a href="#q16-1" class="blo
                        <a href="#q16-2" class="blo
                        <a href="#q16-3" class="blo
                    </div>
                </details>
```

```
                                if (other !== detai
                                    other.removeAtt
                                }
                            });
                        }
                    });
                });

                // Scroll spy logic
                document.addEventListener('DOMConte
                    const sections = document.query
            ="q5-"], div[id^="q6-"], div[id^="q7-"], di
            v[id^="q13-"], div[id^="q14-"], div[id^="q1
                    let isScrolling = false;

                    document.querySelectorAll('.gro
                        link.addEventListener('clic
                            isScrolling = true;
                            setTimeout(() => isScro
                        });
                    });

                    const observer = new Intersecti
                        if (isScrolling) return;

                        let visibleId = null;
                        entries.forEach(entry => {
                            if (entry.isIntersectin
                                visibleId = entry.t

                                // Highlight active
                                document.querySelec
                                    a.style.fontWei
                                    if (a.href.ends
                                        a.style.fon
                                    }
                                });
```

```
                        modal.style.justifyContent = 'c

                        const box = document.createElem
                        box.style.width = '80%';
                        box.style.height = '80%';
                        box.style.backgroundColor = '#1
                        box.style.padding = '20px';
                        box.style.borderRadius = '10px'
                        box.style.display = 'flex';
                        box.style.flexDirection = 'colu
                        box.style.color = '#fff';

                        const header = document.createE
                        header.innerText = 'Экспорт в '
                        header.style.marginBottom = '10

                        const subHeader = document.crea
                        subHeader.innerText = 'Из-за пе
            ние веб-студии в новой вкладке (через меню)
                        subHeader.style.marginBottom =
                        subHeader.style.color = '#f8717
                        subHeader.style.fontSize = '0.9

                        const textarea = document.creat
                        textarea.value = contentRaw;
                        textarea.style.flex = '1';
                        textarea.style.width = '100%';
                        textarea.style.color = '#000';
                        textarea.style.padding = '10px'
                        textarea.style.fontFamily = 'mo

                        const btnContainer = document.c
                        btnContainer.style.display = 'f
                        btnContainer.style.gap = '10px'
                        btnContainer.style.marginTop =

                        const copyBtn = document.create
                        copyBtn.innerText = 'Копировать
```

```
                    }

                    function downloadMD() {
                        let clone = document.querySelec

                        // Smart markdown parsing based

                        // Smart markdown parsing based
                        clone.querySelectorAll('h2').fo
                        clone.querySelectorAll('h3').fo
                        clone.querySelectorAll('.analog
                        clone.querySelectorAll('li').fo
                        clone.querySelectorAll('svg').f
                            let caption = '';
                            if (el.nextElementSibling &
                                caption = el.nextElemen
                            }
                            el.textContent = '\n\n[ Илл
                        });
                        clone.querySelectorAll('.img-ca

                        let text = clone.innerText;
                        text = text.replace(/\n{3,}/g,

                        if (window !== window.top) {
                            fallbackModal(text, 'Markdo
                            return;
                        }
                        const blob = new Blob([text], {
                        const url = URL.createObjectURL
                        const link = document.createEle
                        link.href = url;
                        link.download = 'vyisshaya_alge
                        document.body.appendChild(link)
                        link.click();
                        document.body.removeChild(link)
                    }
```

```
                                body.bg-white [class*="
                                body.bg-white .analogy-
                                body.bg-white .img-plac
                                body.bg-white .info-blo

                                body.bg-white .analogy-
          or: #94a3b8 !important; }
                                body.bg-white .img-capt

                                /* Tweak card blocks */
                                body.bg-white .card-red

                                body.bg-white aside { b
           #334155 !important; }
                                body.bg-white .text-whi

                                /* Ensure active links
                                body.bg-white aside a.t
                                body.bg-white aside a.t
                                body.bg-white aside a.t
                                body.bg-white aside a.t
                                body.bg-white aside a.t
                                body.bg-white aside a.t
                                body.bg-white aside a:h

                                body.bg-white table tr.
                                body.bg-white table tr.
                                body.bg-white .text-tea
                                body.bg-white .text-ros
                                body.bg-white .text-eme
                                body.bg-white .text-cya
                                body.bg-white .text-amb
                                body.bg-white .text-fuc
                                body.bg-white .text-lim
                                body.bg-white .text-blu
                                body.bg-white .text-gre
                                body.bg-white .text-ind
```

```
            -sm font-bold flex items-center justify-cen
                            <span> </span> PDF
                        </a>
                        <button onclick="downloadMD()" 
            -bold flex items-center justify-center gap-
                            <span> </span> Markdown
                        </button>
                        <button onclick="downloadHTML()
            t-sm font-bold flex items-center justify-ce
                            <span> </span> HTML
                        </button>
                    </div>

                    <!-- Смена темы -->
                    <div>
                        <div class="text-xs text-slate-
                        <div class="flex items-center g
                            <button id="theme-dark" onc
            justify-center text-amber-300 hover:text-wh
            line-amber-500" title="Темная тема"> </butt
                            <button id="theme-light" on
            r justify-center text-amber-50 hover:text-w
            ">☀ </button>
                            <button id="theme-terminal"
            -center justify-center text-green-400 hover
             терминала"> </button>
                        </div>
                    </div>
                </div>

            </div>
        </aside>

        <!-- Основной контент -->
        <main class="flex-1 bg-slate-800 p-4 sm:p-8 md:
          -base min-w-0">
            <header class="text-center mb-12 border-b b
                <h1 class="text-4xl md:text-5xl font-bl
```

```
            <div class="overflow-x-auto w-full pb-4
                <div class="grid grid-cols-3 gap-4 
                    <!-- Column headers -->
                    <div class="bg-yellow-900/40 bo
        оля и кольца, целые числа</div>
                    <div class="bg-orange-900/40 bo
        ольцо полиномов</div>
                    <div class="bg-cyan-900/40 bord
        сные числа</div>

                <!-- Row 1: База -->
                <a href="#question-1" onclick="setM
        -4 border-yellow-500 hover:bg-slate-700 tra
                    <div class="text-xs text-yellow
                    <div class="font-bold text-slat
                </a>
                <a href="#question-9" onclick="setM
        -4 border-orange-500 hover:bg-slate-700 tra
                    <div class="text-xs text-orange
                    <div class="font-bold text-slat
                </a>
                <a href="#question-2" onclick="setM
        -4 border-cyan-500 hover:bg-slate-700 trans
                    <div class="text-xs text-cyan-5
                    <div class="font-bold text-slat
                </a>

                <!-- Row 2: Базовые операции / Евкл
                <a href="#question-6" onclick="setM
        -4 border-yellow-500 hover:bg-slate-700 tra
                    <div class="text-xs text-yellow
                    <div class="font-bold text-slat
                </a>
                <a href="#question-10" onclick="set
        -l-4 border-orange-500 hover:bg-slate-700 t
                    <div class="text-xs text-orange
                    <div class="font-bold text-slat
```

```
                <!-- Row 5: Корни / Уравнения -->
                <div class="hidden md:block"></div>
                <a href="#question-13" onclick="setM
        -l-4 border-orange-500 hover:bg-slate-700 t
                    <div class="text-xs text-orange
                    <div class="font-bold text-slat
                </a>
                <a href="#question-4" onclick="setM
        -4 border-cyan-500 hover:bg-slate-700 trans
                    <div class="text-xs text-cyan-5
                    <div class="font-bold text-slat
                </a>

                <!-- Row 6: Кратность -->
                <div class="hidden md:block"></div>
                <a href="#question-14" onclick="setM
        -l-4 border-orange-500 hover:bg-slate-700 t
                    <div class="text-xs text-orange
                    <div class="font-bold text-slat
                </a>
                <div class="hidden md:block"></div>

                <!-- Row 7: Производная -->
                <div class="hidden md:block"></div>
                <a href="#question-15" onclick="setM
        -l-4 border-orange-500 hover:bg-slate-700 t
                    <div class="text-xs text-orange
                    <div class="font-bold text-slat
                </a>
                <div class="hidden md:block"></div>

                <!-- КЛАСТЕР 3: НЕПРИВОДИМОСТЬ -->
                <div class="col-span-1 md:col-span-1
        b-2 border-b border-emerald-500/30 pb-2">Кла

                <a href="#question-17" onclick="setM
        -l-4 border-teal-500 hover:bg-slate-700 tra
                    <div class="text-xs text-teal-5
```

```
                </div>
                </div>

                <!-- ЛИКБЕЗ SECTION INSERTED HERE -->
                <section id="proof-algorithms" class="m
                    <div class="flex items-center mb-6"
                        <div class="bg-red-500/20 p-3 r
                            <svg class="w-8 h-8 text-re
            rg/2000/svg">
                                <path stroke-linecap="r
             110 4m-6 8a2 2 0 100-4m0 4a2 2 0 110-4m0 4
                            </svg>
                        </div>
                        <h2 class="text-3xl pr-4 border
                    </div>

                    <div class="prose prose-invert max-
                        <p class="text-lg">
                            Хватит тупить. Доказательст
            есь для того, чтобы жать на кнопки. Вот жес
                        </p>

                        <h2 class="text-2xl font-bold b
            (Меметика)</h2>

                        <!-- Прямое -->
                        <div class="bg-slate-800/80 p-6
                            <h3 class="text-xl text-blu
                            <p class="text-sm text-slat
                            <ul class="list-disc pl-5 s
                                <li><span class="text-w
            k$).</li>
                                <li><span class="text-w
                                <li><span class="text-w
            ово.</li>
                            </ul>
                        </div>
```

```
                    </ul>
                </div>

                <!-- Двойная делимость -->
                <div class="bg-slate-800/80 p-6
                    <h3 class="text-xl text-ora
                    <p class="text-sm text-slat
                    <ul class="list-disc pl-5 s
                        <li><span class="text-w
 вверх по уравнениям).</li>
                        <li><span class="text-w
рху вниз, показывая, что $\delta \vdots r_i
                        <li><span class="text-w
х. Твой $d$ больше.</li>
                    </ul>
                </div>


                <h2 class="text-2xl font-bold b
оритмы экзекуции)</h2>

                <div class="grid grid-cols-1 md
                    <div class="bg-slate-800 p-
                        <h4 class="text-red-400
                        <p class="text-sm text-
                        <p class="text-xs font-
со следующим. Если в конце 0 — пациент мерт
                    </div>

                    <div class="bg-slate-800 p-
                        <h4 class="text-red-400
                        <p class="text-sm text-
                        <p class="text-xs font-
$c$. Пока получается 0 — корень жив. Как то
корня минус один.</p>
                    </div>

                    <div class="bg-slate-800 p-
```

```
                                <code class="text-x
                            </div>
                        </li>

                        <li class="bg-slate-800/50 
                            <span class="text-2xl">
                            <div>
                                <h4 class="text-cya
                                <p class="text-sm t
            есть сложение — есть ноль. Если есть умноже
                                <code class="text-x
                            </div>
                        </li>

                        <li class="bg-slate-800/50 
                            <span class="text-2xl">
                            <div>
                                <h4 class="text-cya
                                <p class="text-sm t
            авь их драться. Умножение всегда врывается 
                                <code class="text-x
                            </div>
                        </li>
                    </ul>
                </div>
            </section>
        </div>

        <div id="questions-container">
        <!-- ═══════════════════ ВОПРОС 1 ═══════════
```

РАЗДЕЛ: УТИЛИТЫ

```
await build({
  entryPoints: [path.join(ROOT, "src/data/content.ts")]
  bundle: true,
  format: "esm",
  platform: "node",
  outfile: bundle,
  logLevel: "error",
});

const { chapters } = await import(`file://${bundle}`);

// Список интерактивов берём из реестра визуализаторов
const registry = fs.readFileSync(path.join(ROOT, "src/c
const registryBody = registry.slice(registry.indexOf("V
const vizIds = new Set([...registryBody.matchAll(/^\s{2

const rows = chapters.map((c) => {
  const html = c.content ?? "";
  const analogies = (html.match(/Аналогия/gi) ?? []).le
  return {
    id: c.id,
    title: c.title,
    num: Number.parseInt(c.title.match(/^(\d+)\./)?.[1]
    analogies,
    hasAnalogy: analogies > 0,
    inlineSvg: /<svg[\s>]/i.test(html),
    image: /<img[\s>]/i.test(html),
    interactive: vizIds.has(c.id),
    chars: html.length,
  };
});

const has2D = (r) => r.interactive || r.inlineSvg || r.
const gapBoth = rows.filter((r) => !r.hasAnalogy && !ha
const gapAnalogy = rows.filter((r) => !r.hasAnalogy && !
const gapViz = rows.filter((r) => r.hasAnalogy && !has2
const orphanViz = [...vizIds].filter((id) => !chapters.
```

```
 * Склеивает отрендеренные страницы tmp/export-pages/pa
 * в единый документ public/export/full_code_all_pages.p
 */
import fs from "node:fs";
import path from "node:path";
import PDFDocument from "pdfkit";
import { ROOT, OUT_DIR, PAGES_DIR, PDF_PATH, ensureDir,

// A4 в пунктах PDF
const A4 = { width: 595.28, height: 841.89 };

async function main() {
  if (!fs.existsSync(PAGES_DIR)) {
    throw new Error(`Нет ${path.relative(ROOT, PAGES_DI
  }

  const pages = fs
    .readdirSync(PAGES_DIR)
    .filter((f) => f.endsWith(".png"))
    .sort();

  if (pages.length === 0) throw new Error("Не найдено н

  ensureDir(OUT_DIR);

  const doc = new PDFDocument({
    size: [A4.width, A4.height],
    margin: 0,
    autoFirstPage: false,
    info: {
      Title: "Универсальное пособие — полный исходный к
      Author: "CI: rebuild-pdf",
      Subject: "Экспорт всего кода проекта (код -> txt
      CreationDate: new Date(),
    },
  });

  const stream = fs.createWriteStream(PDF_PATH);
```

Универсальное пособие • полный исходный код

```
import fs from "node:fs";
import path from "node:path";
import { execFileSync } from "node:child_process";
import { ROOT, OUT_DIR, TXT_PATH, HR, SUBHR, ensureDir,

/** Расширения, которые считаем «кодом» и включаем в да
const CODE_EXT = new Set([
  ".ts", ".tsx", ".js", ".jsx", ".mjs", ".cjs",
  ".css", ".scss", ".html", ".json", ".md", ".yml", ".y
]);

/** Явные исключения: генерируемое, бинарное и просто ш
const EXCLUDE_FILES = new Set(["package-lock.json", "bu
const EXCLUDE_DIRS = ["node_modules", "dist", "build",

/** Человекочитаемые категории для оглавления. */
const CATEGORIES = [
  [/^src\/data\/tickets\//, "Билеты (учебный контент)"]
  [/^src\/data\//, "Контент и данные пособия"],
  [/^src\/components\/visualizers\//, "Визуализаторы (в
  [/^src\/components\//, "Интерактивные компоненты и си
  [/^src\/layout\//, "HTML-разметка (layout)"],
  [/^src\/utils\//, "Утилиты"],
  [/^src\//, "Ядро приложения"],
  [/^scripts\//, "Скрипты сборки и экспорта"],
  [/^\.github\//, "CI / автоматизация"],
  [/^[^/]+$/, "Конфигурация проекта"],
];

const categoryOf = (rel) => CATEGORIES.find(([re]) => r

/** Порядок разделов в документе. */
const ORDER = [
  "Конфигурация проекта",
  "Ядро приложения",
  "Контент и данные пособия",
  "Билеты (учебный контент)",
  "Интерактивные компоненты и симуляторы",
```

```
      const ib = ORDER.indexOf(categoryOf(b));
      return ia !== ib ? ia - ib : a.localeCompare(b);
    });
}

/** Достаём заголовки глав из данных пособия (без запус
function extractChapters(files) {
  const chapters = [];
  const seen = new Set();
  for (const rel of files.filter((f) => f.startsWith("s
    const src = fs.readFileSync(path.join(ROOT, rel), "
    const re = /id:\s*"([^"]+)"\s*,\s*\n?\s*title:\s*"(
    let m;
    while ((m = re.exec(src)) !== null) {
      const [, id, title] = m;
      if (seen.has(id)) continue;
      seen.add(id);
      chapters.push({ id, title, file: rel });
    }
  }
  return chapters.sort((a, b) => {
    const na = parseInt(a.title.match/(\d+)/)?.[0] ??
    const nb = parseInt(b.title.match/(\d+)/)?.[0] ??
    return na - nb;
  });
}

function main() {
  const files = listFiles();
  const chapters = extractChapters(files);
  ensureDir(OUT_DIR);

  const out = [];
  const push = (s = "") => out.push(s);

  const stamp = new Date().toLocaleString("ru-RU", { ti
  const totalBytes = files.reduce((s, f) => s + fs.stat
```

```
----------------------------------------------------------------
    if (cat !== current) {
      current = cat;
      push(HR);
      push(`РАЗДЕЛ: ${cat.toUpperCase()}`);
      push(HR);
      push();
    }
    const content = fs.readFileSync(path.join(ROOT, rel
    const lines = content.split("\n");
    push(SUBHR);
    push(`ФАЙЛ ${i + 1}/${files.length}: ${rel}`);
    push(`КАТЕГОРИЯ: ${cat} | СТРОК: ${lines.length} | 
    push(SUBHR);
    push();
    push(content.replace(/\n+$/, ""));
    push();
    push();
  });

  push(HR);
  push(`КОНЕЦ ДОКУМЕНТА • ${files.length} файлов • сген
  push(HR);

  const txt = out.join("\n") + "\n";
  fs.writeFileSync(TXT_PATH, txt, "utf8");

  console.log("[1/3] код -> txt");
  console.log(`      файлов:  ${files.length}`);
  console.log(`      глав:    ${chapters.length}`);
  console.log(`      строк:   ${txt.split("\n").length}
  console.log(`      выход:   ${path.relative(ROOT, TXT_
}

main();


----------------------------------------------------------------
ФАЙЛ 72/74: scripts/export/common.mjs
```

```
----------------------------------------------------------------
  rows: 76,
};

export const HR = "=".repeat(PAGE.cols);
export const SUBHR = "-".repeat(PAGE.cols);

export function ensureDir(dir) {
  fs.mkdirSync(dir, { recursive: true });
}

export function humanSize(bytes) {
  if (bytes < 1024) return `${bytes} B`;
  if (bytes < 1024 * 1024) return `${(bytes / 1024).toF
  return `${(bytes / 1024 / 1024).toFixed(2)} MB`;
}

/** ImageMagick 7 (`magick`) или 6 (`convert`). */
export function imageMagickCmd() {
  for (const cmd of ["magick", "convert"]) {
    try {
      execFileSync(cmd, ["-version"], { stdio: "ignore"
      return cmd;
    } catch {
      /* пробуем следующий */
    }
  }
  throw new Error(
    "ImageMagick не найден. Установите его: sudo apt-ge
  );
}


----------------------------------------------------------------
ФАЙЛ 73/74: scripts/export/render-pages.mjs
КАТЕГОРИЯ: Скрипты сборки и экспорта | СТРОК: 117 | РАЗ
----------------------------------------------------------------

/**
```

```
  for (let i = 0; i < source.length; i += PAGE.rows) {
    pages.push(source.slice(i, i + PAGE.rows));
  }
  return pages;
}

async function main() {
  if (!fs.existsSync(TXT_PATH)) {
    throw new Error(`Нет ${path.relative(ROOT, TXT_PATH
  }

  const im = imageMagickCmd();
  const pages = paginate(fs.readFileSync(TXT_PATH, "utf

  fs.rmSync(PAGES_DIR, { recursive: true, force: true }
  ensureDir(PAGES_DIR);

  const total = pages.length;
  const jobs = pages.map((lines, idx) => {
    const num = String(idx + 1).padStart(4, "0");
    const header = `Универсальное пособие • полный исхо
      Math.max(1, PAGE.cols - 44 - `стр. ${idx + 1} / $
    )}стр. ${idx + 1} / ${total}`;
    const body = [header, "-".repeat(PAGE.cols), ...lin
    const txtFile = path.join(PAGES_DIR, `page-${num}.t
    const pngFile = path.join(PAGES_DIR, `page-${num}.p
    fs.writeFileSync(txtFile, body + "\n", "utf8");
    return { txtFile, pngFile, num };
  });

  const args = (job) => [
    "-density", String(PAGE.density),
    "-page", PAGE.size,
    "-font", fs.existsSync(PAGE.fontFile) ? PAGE.fontFi
    "-pointsize", String(PAGE.pointsize),
    `text:${job.txtFile}`,
    "-background", "white",
    "-flatten",
```

```
----------------------------------------------------------------
});


================================================================
РАЗДЕЛ: CI / АВТОМАТИЗАЦИЯ
================================================================


----------------------------------------------------------------
ФАЙЛ 74/74: .github/workflows/rebuild-pdf.yml
КАТЕГОРИЯ: CI / автоматизация | СТРОК: 128 | РАЗМЕР: 4.
----------------------------------------------------------------

name: Rebuild code PDF

# Пайплайн пересборки PDF при каждом коммите:
#
#   исходный код  ──►  full_code_all_pages.txt   (script
#        │
#        └──────────►  page-0001.png … page-NNNN.png  (
#                           │
#                           └──►  full_code_all_page
#
# Готовые TXT и PDF коммитятся обратно в ветку (их отда
# промежуточные PNG сохраняются как artifacts запуска.

on:
  push:
    branches:
      - "**"
    paths-ignore:
      # чтобы коммит самого workflow не запускал бескон
      - "public/export/**"
      - "**/*.md"
  workflow_dispatch:
    inputs:
      density:
        description: "DPI растеризации страниц (по умол
        required: false
```

```
            fi
          done
          convert -version

      - name: Install dependencies
        run: npm ci --no-audit --no-fund

      - name: Step 1 — код → txt
        run: npm run export:txt

      - name: Step 2 — txt → png
        env:
          EXPORT_DENSITY: ${{ github.event.inputs.densi
        run: npm run export:png

      - name: Step 3 — png → pdf
        run: npm run export:pdf

      - name: Type-check приложения
        run: npx tsc --noEmit
        continue-on-error: true

      - name: Summary
        run: |
          {
            echo "###   Экспорт пересобран"
            echo ""
            echo "| Артефакт | Размер |"
            echo "| --- | --- |"
            echo "| \`public/export/full_code_all_pages
            echo "| \`public/export/full_code_all_pages
            echo "| PNG-страниц | $(ls tmp/export-pages.
          } >> "$GITHUB_STEP_SUMMARY"

      - name: Upload artifacts (PDF + TXT)
        uses: actions/upload-artifact@v4
        with:
          name: full-code-export
```