ПРИМЕНЕНИЕ АЛГОРИТМА ДЕЙКСТРЫ ДЛЯ ПОСТРОЕНИЯ ТРАЕКТОРИИ ДВИЖЕНИЯ В 2D-ПРОСТРАНСТВЕ

АЛЕКСАНДР ПОНОМАРЕНКО

ГК Иннотех, Frontend-разработчик React.

В web-разработке 20 лет, фронтенд – 4 года.

Опыт создания обучающих игр на JS

Telegram: https://t.me/brain16383

СОДЕРЖАНИЕ

1. АЛГОРИТМ ДЕЙКСТРЫ

Алгоритм Дейкстры — алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных

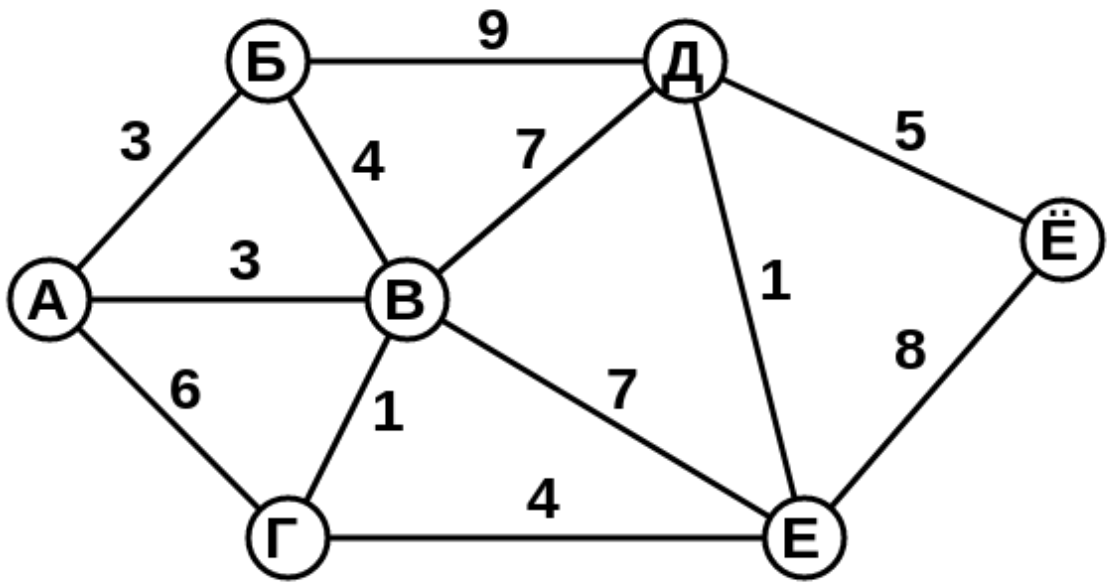
ГДЕ МОЖЕТ ПРИГОДИТЬСЯ АЛГОРИТМ ДЕЙКСТРЫ?

- Маршрутизация в компьютерных сетях
- Программы-навигаторы: поиск маршрута, чтобы быстрее добраться из точки А в точку Б
- В играх (вычисление оптимальной траектории движения)
- Расчет электрических цепей: ток течет по пути наименьшего сопротивления
- ...

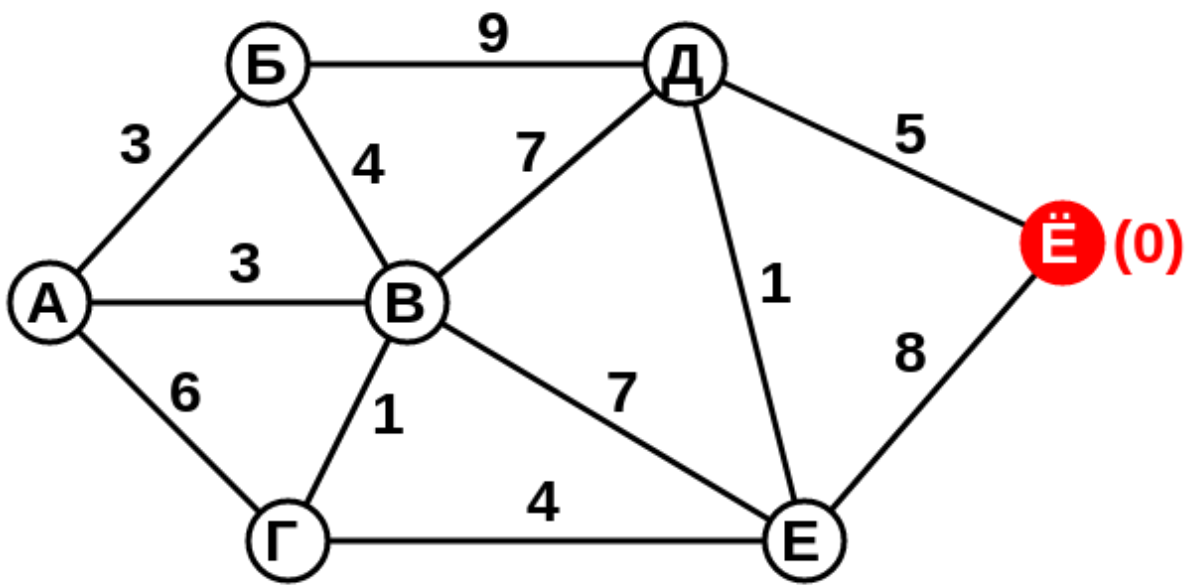
1. АЛГОРИТМ ДЕЙКСТРЫ. ОСНОВНЫЕ МОМЕНТЫ

- Внешний цикл - обходим все вершины графа
- Внутренний цикл - обходим соседние вершины относительно текущей
- Расчет стоимости перехода в соседнюю вершину из текущей
- При расчете стоимости запоминаем, по какому ребру перешли в данную вершину

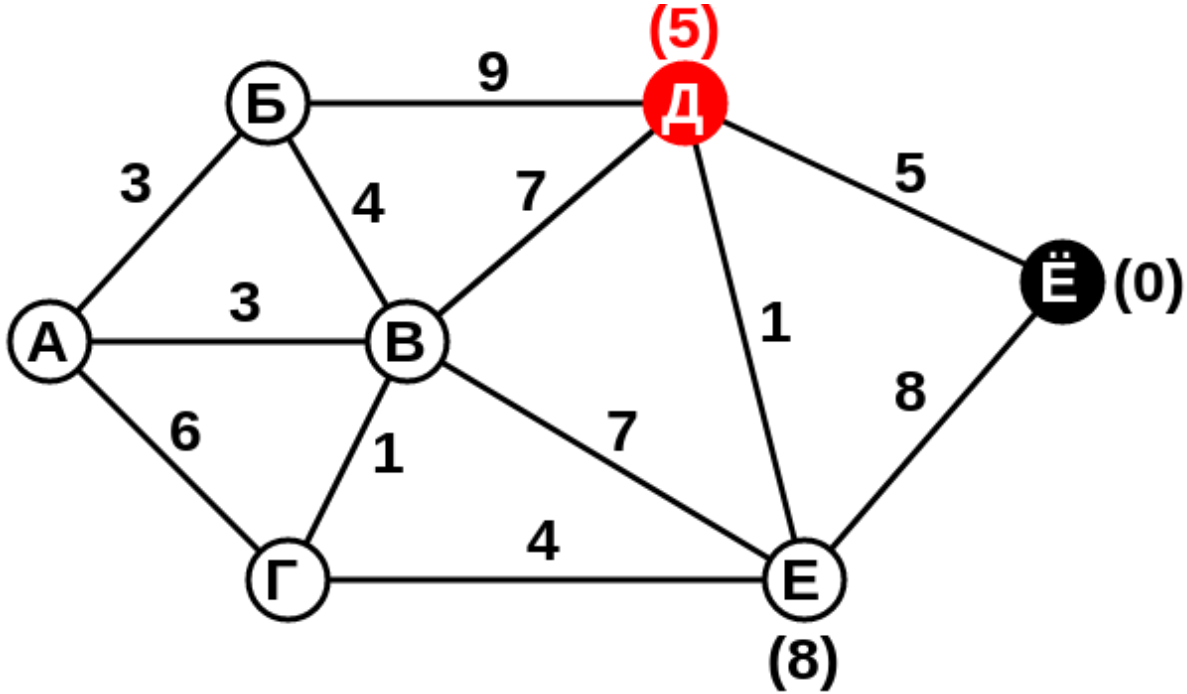
1. АЛГОРИТМ ДЕЙКСТРЫ. ИСХОДНЫЙ ГРАФ



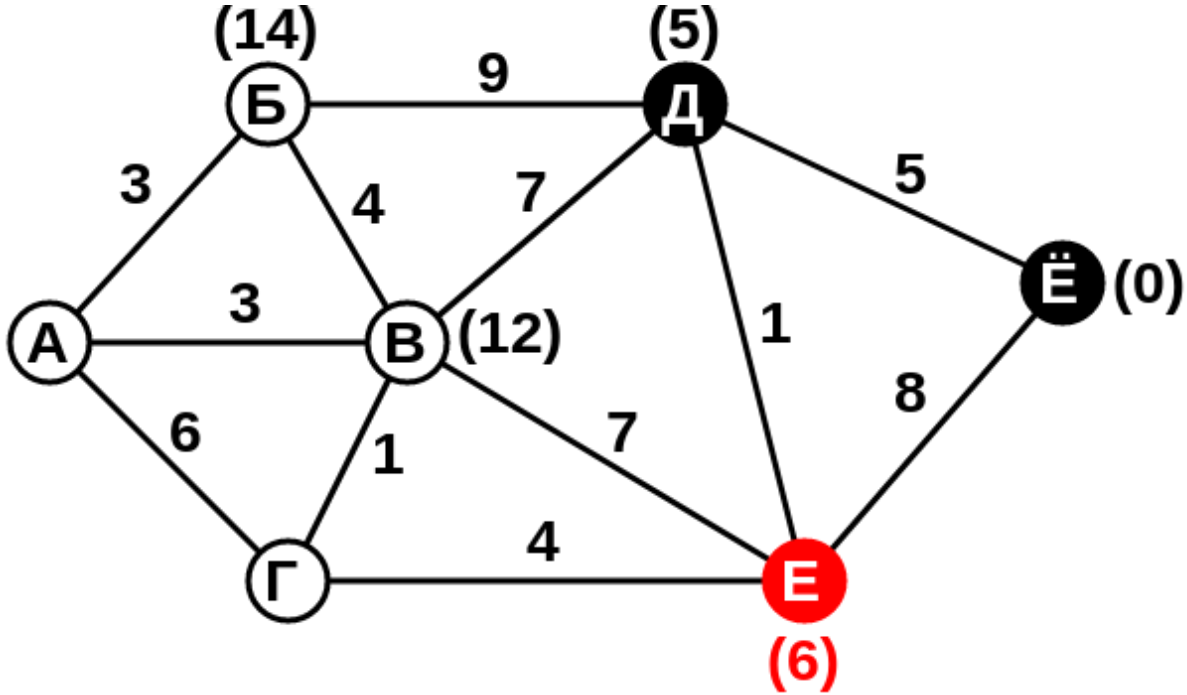
1. АЛГОРИТМ ДЕЙКСТРЫ. ШАГ 0



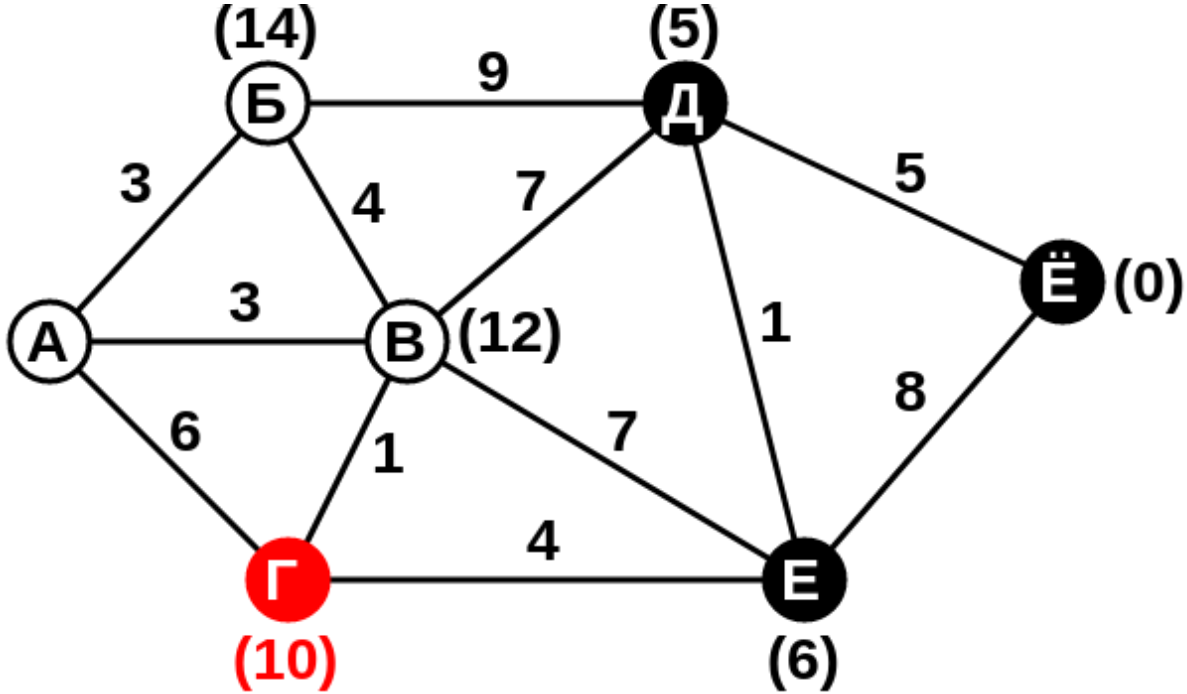
1. АЛГОРИТМ ДЕЙКСТРЫ. ШАГ 1



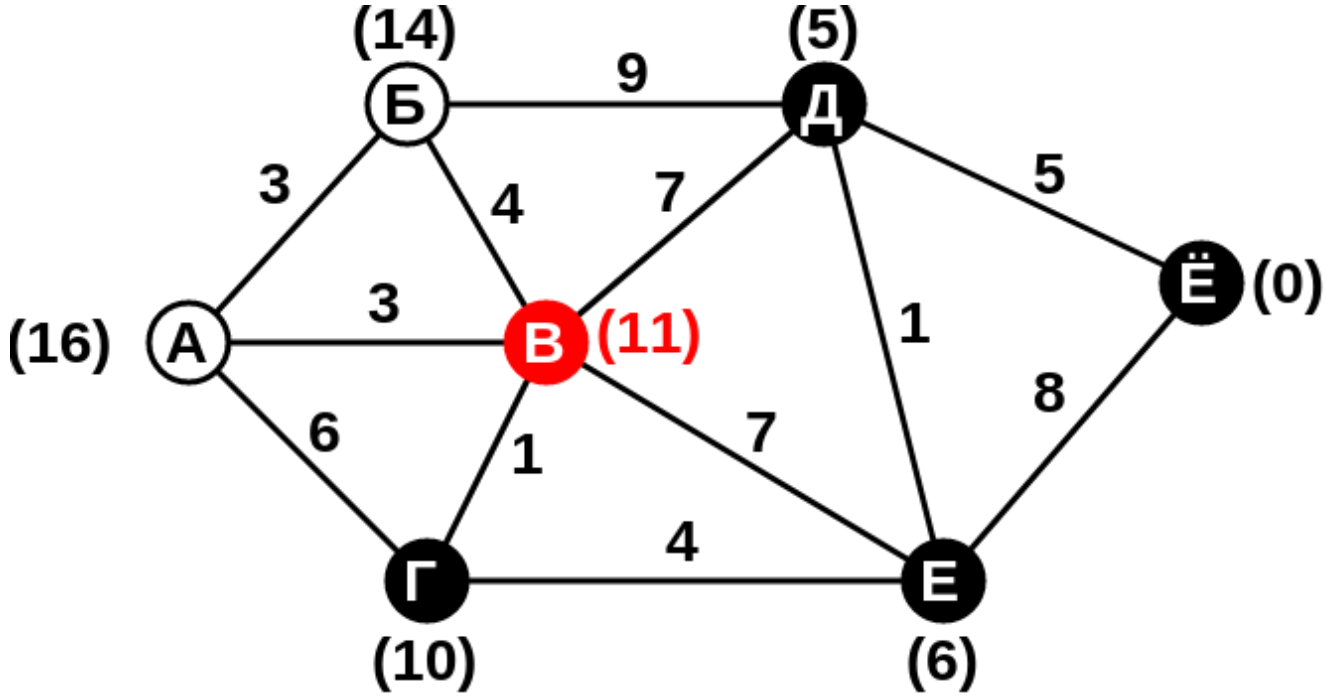
1. АЛГОРИТМ ДЕЙКСТРЫ. ШАГ 2



1. АЛГОРИТМ ДЕЙКСТРЫ. ШАГ 3

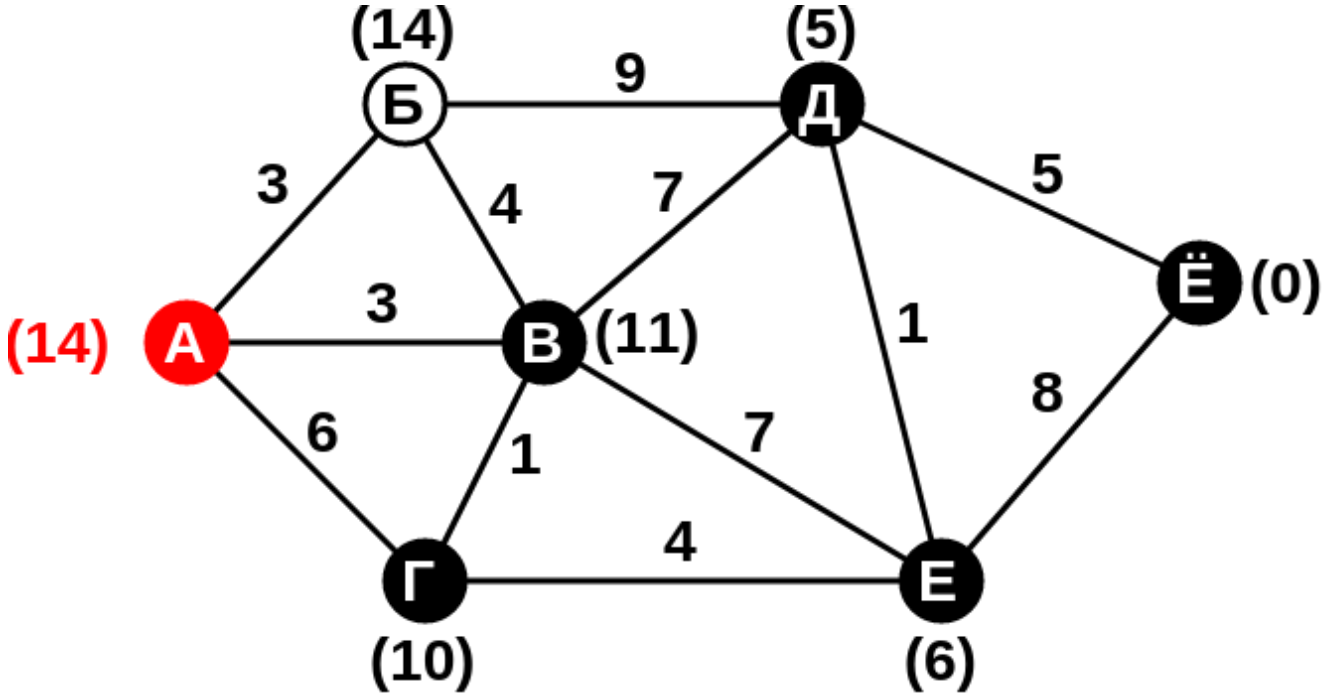


1. АЛГОРИТМ ДЕЙКСТРЫ. ШАГ 4



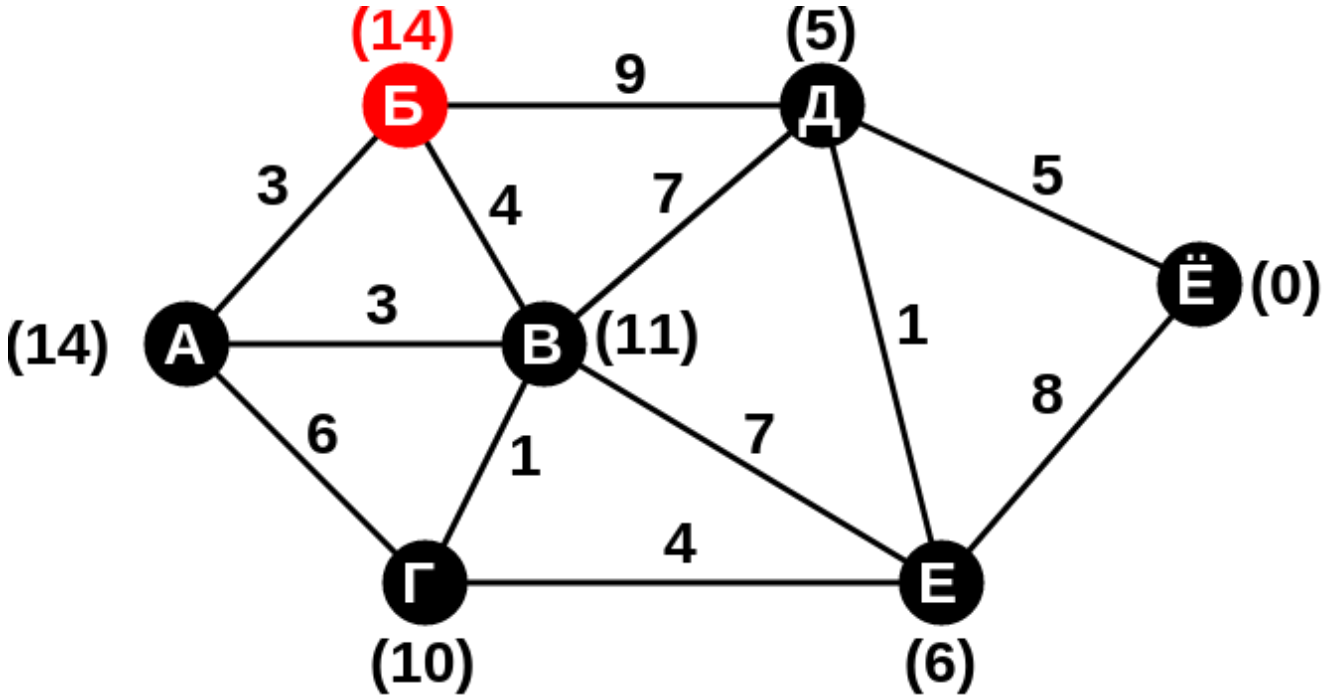
1. АЛГОРИТМ ДЕЙКСТРЫ. ШАГ 5

1. АЛГОРИТМ ДЕЙКСТРЫ. ШАГ 5

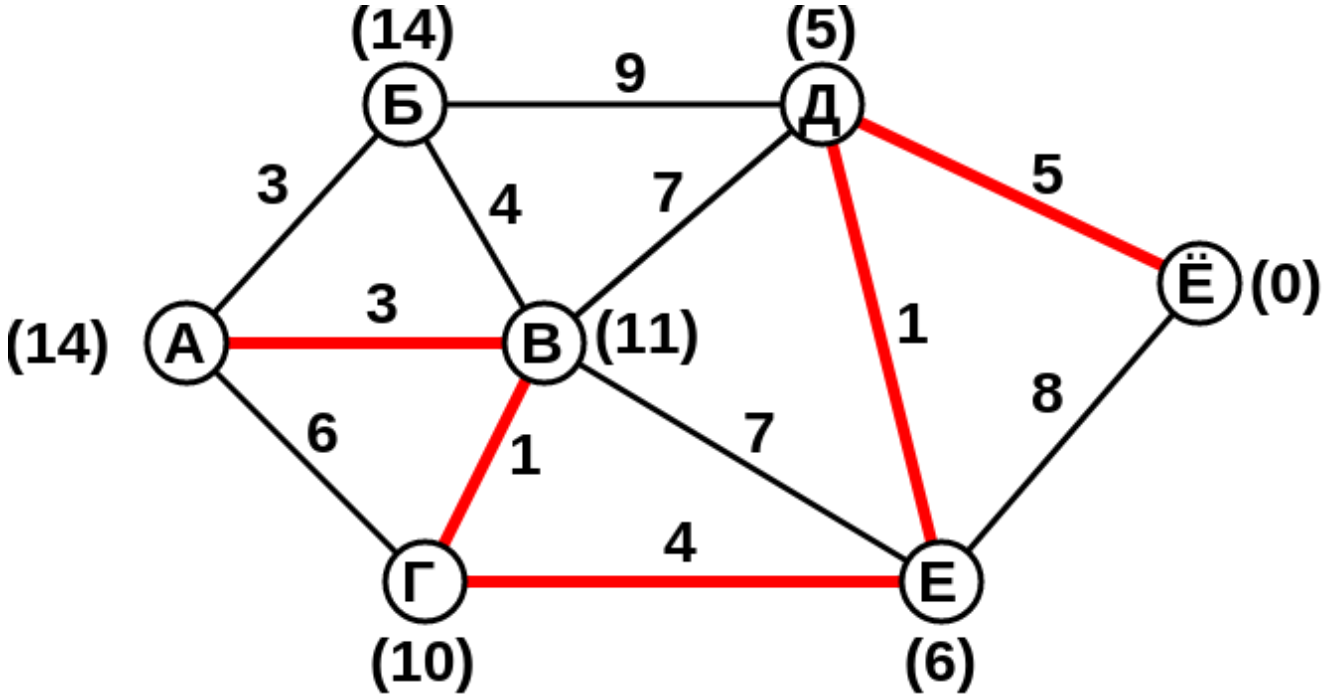


1. АЛГОРИТМ ДЕЙКСТРЫ. ШАГ 6

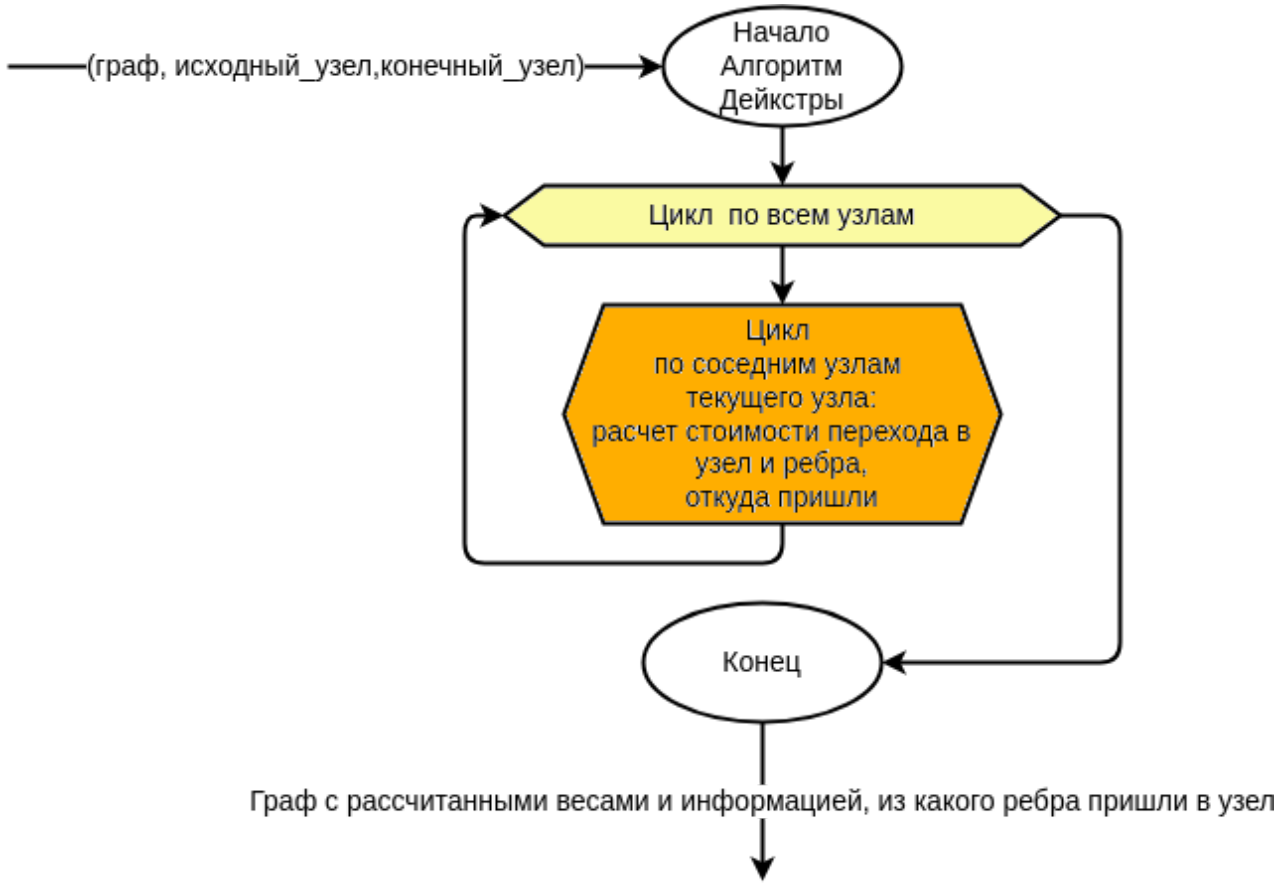
1. АЛГОРИТМ ДЕЙКСТРЫ: ШАГ 6



1. АЛГОРИТМ ДЕЙКСТРЫ: РАССЧИТАН КРАТЧАЙШИЙ ПУТЬ



1. АЛГОРИТМ ДЕЙКСТРЫ


(граф, исходный_узел,конечный_узел)
Начало
Алгоритм
Дейкстры
Цикл по всем узлам
Цикл
по соседним узлам
текущего узла:
расчет стоимости перехода в
узел и ребра,
откуда пришли
Конец
Граф с рассчитанными весами и информацией, из какого ребра пришли в узел

1. ВНЕШНИЙ И ВНУТРЕННИЙ ЦИКЛЫ АЛГОРИТМА


```
01. calcVerticesCost = (fromVertex: number) => {
02.     this.vertices[fromVertex].accessCost = 0;
03.     this.curVertexIndex = fromVertex;
04.     while (this.curVertexIndex !== -1) {
05.         const curVertex = this.vertices[this.curVertexIndex];
06.         this.vertices[this.curVertexIndex].processed = true;
07.         const edgesOfVertex = this.getEdgesOfVertex();
08.         for (let i = 0; i < edgesOfVertex.length; i++) { ... }
09.         const nextVertex = this.getNextVertex(edgesOfVertex);
10.         this.curVertexIndex = nextVertex;
11.     }
12. };
```

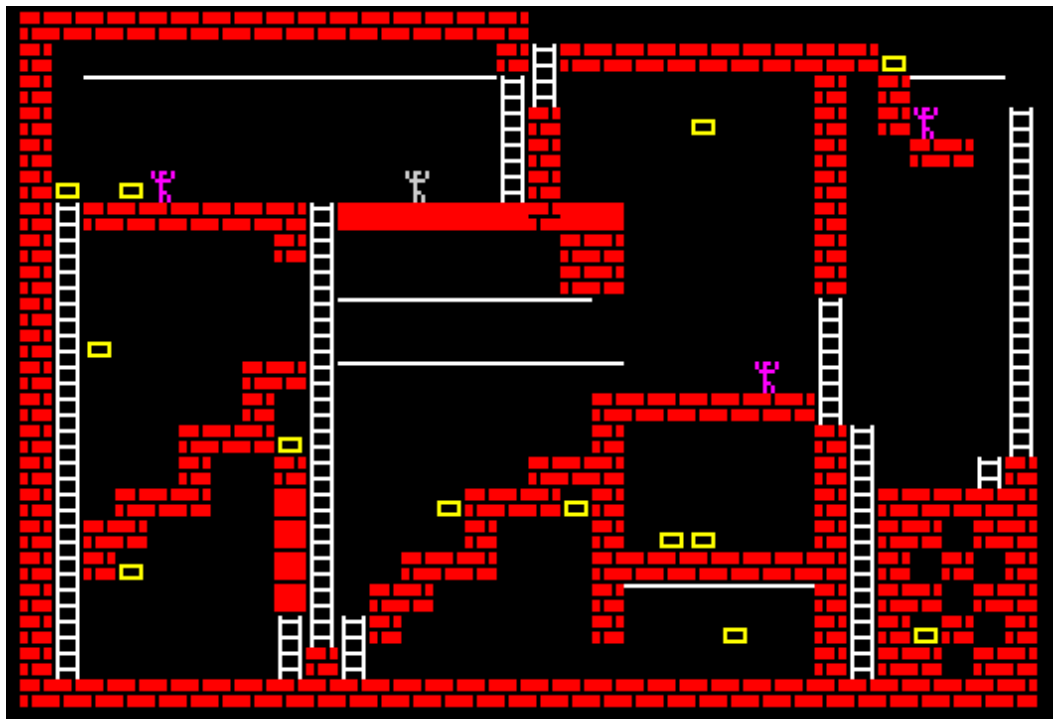
1. ВНУТРЕННИЙ ЦИКЛ АЛГОРИТМА


```
01. for (let i = 0; i < edgesOfVertex.length; i++) {
02.     const edgeIndex = edgesOfVertex[i];
03.     const adjVIndex = this.getAdjancedVIndex(edgeIndex);
04.     const adjVertex = this.vertices[adjacentVIndex];
05.     if (adjacentVertex.processed) {
06.         continue;
07.     }
08.     this.updateAccCostAndEdgeIdx(adjVertex, curVertex,
09.         edgeIndex);
10. }
```

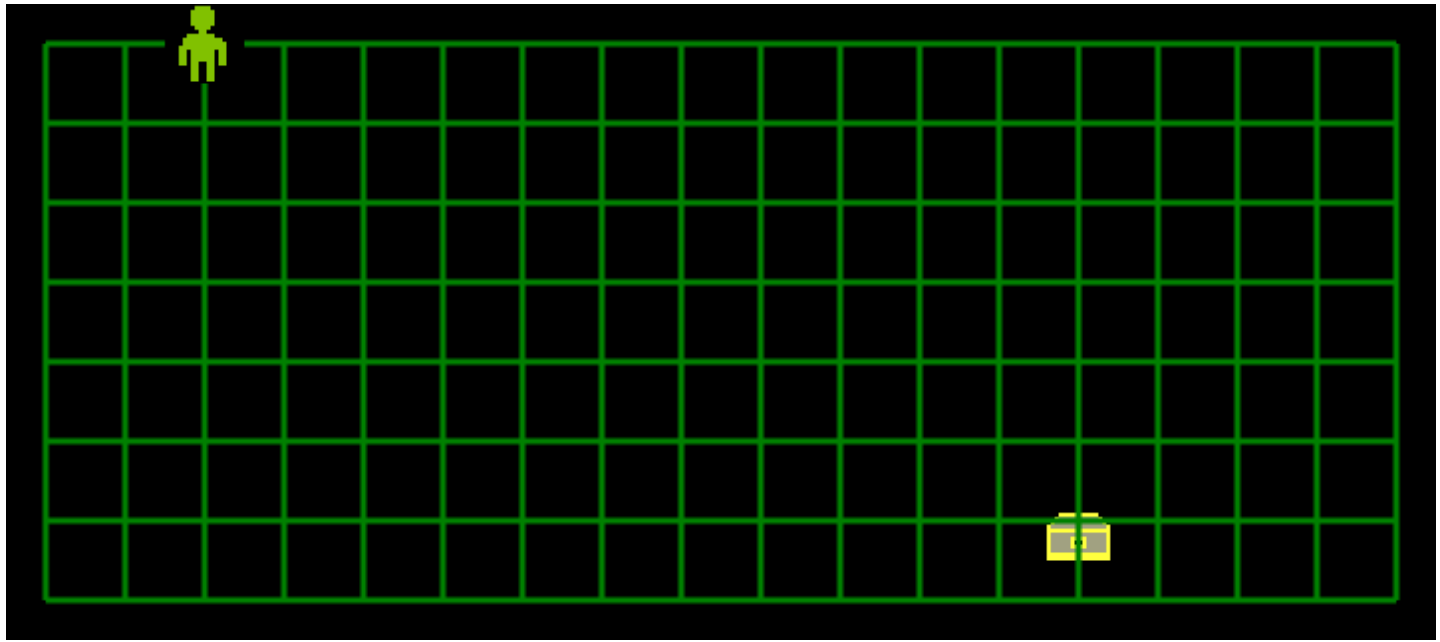
СОДЕРЖАНИЕ

2. ИСТОЧНИК ВДОХНОВЕНИЯ

Игра «Lode runner»
для ZX Spectrum
Broderbund 1984

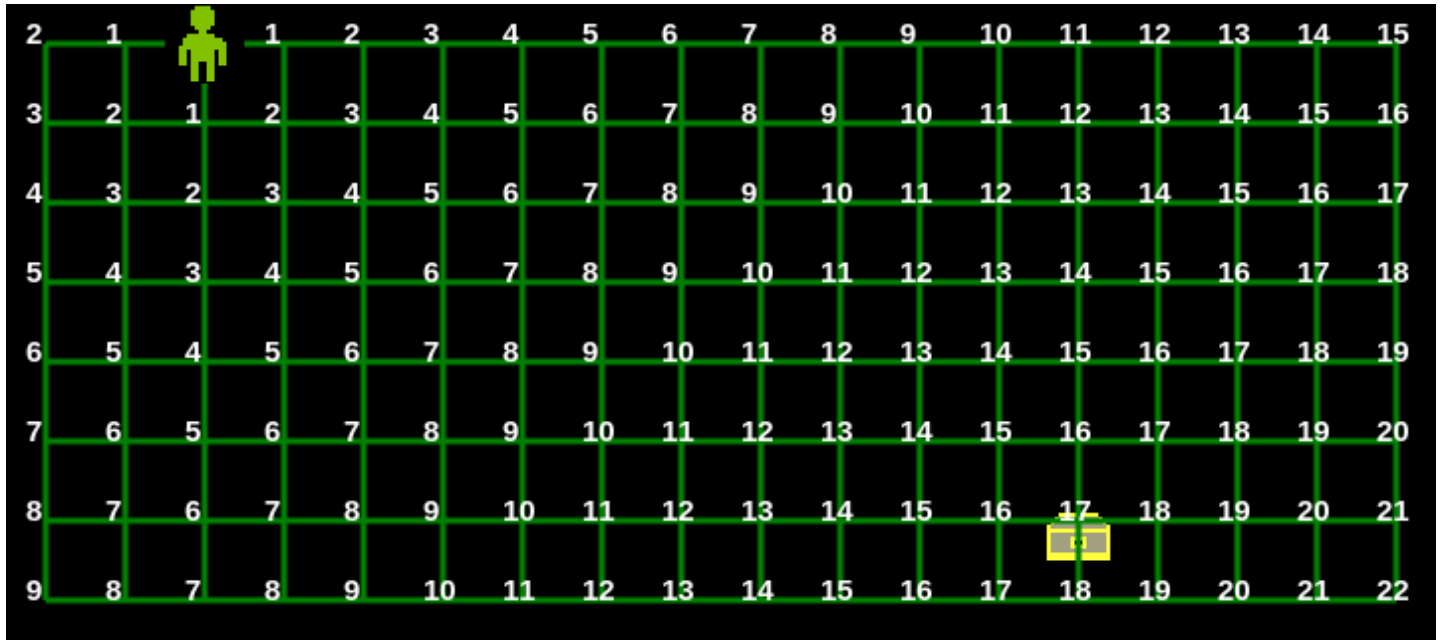


2. КВАДРАТНАЯ СЕТКА



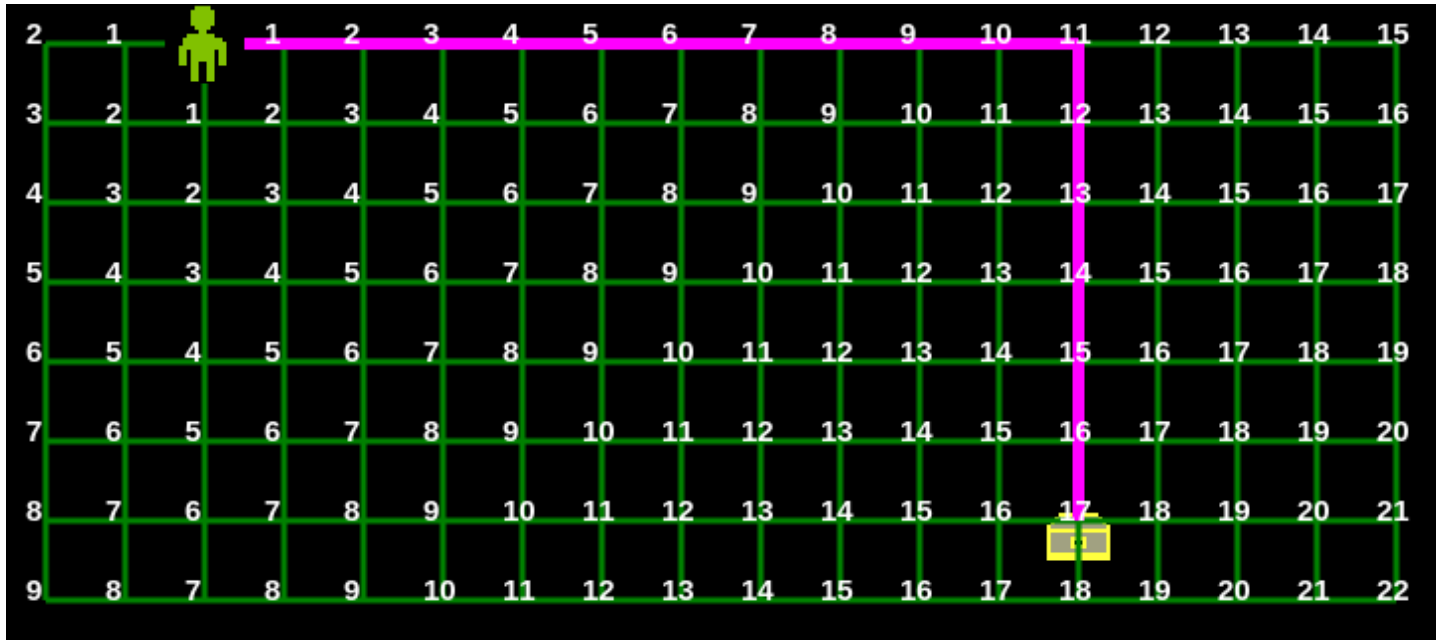
☐ Узлы ☑ Ребра ☐ Путь ☐ Стоимость Старт Стоп

2. КВАДРАТНАЯ СЕТКА



☐ Узлы ☑ Ребра ☐ Путь ☑ Стоимость Старт Стоп

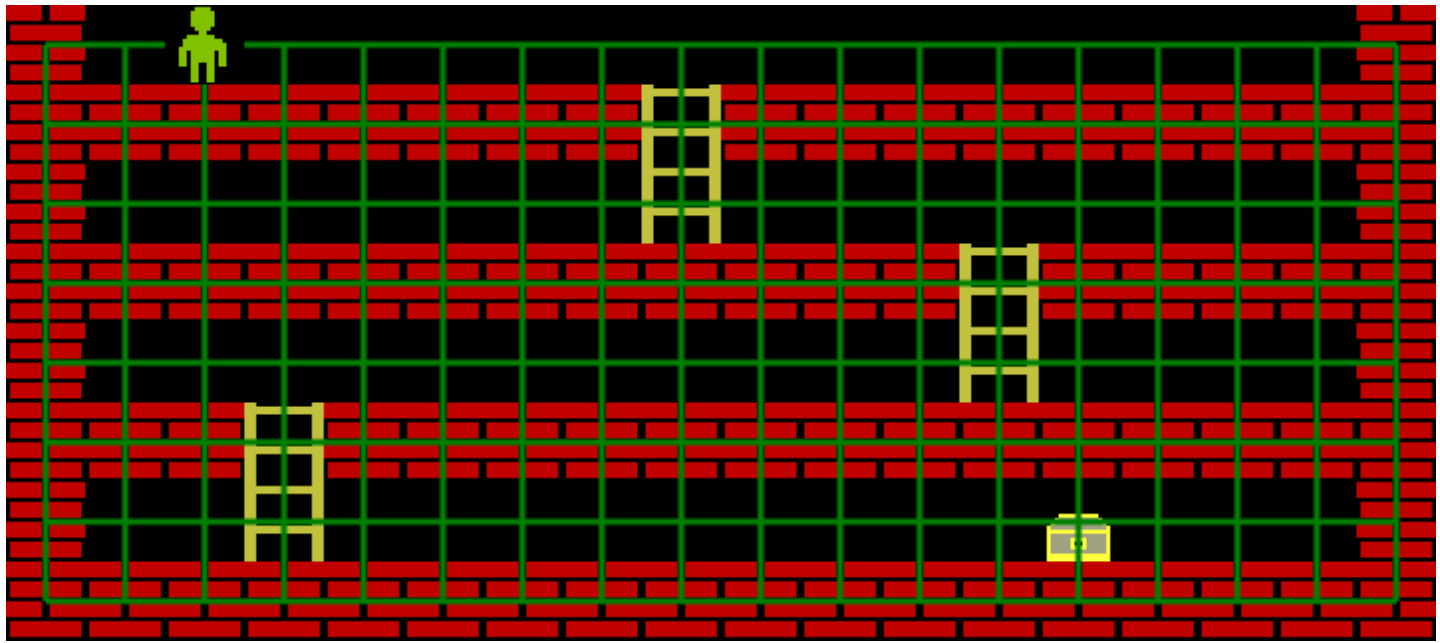
2. КВАДРАТНАЯ СЕТКА



☐ Узлы ☑ Ребра ☑ Путь ☑ Стоимость Старт Стоп

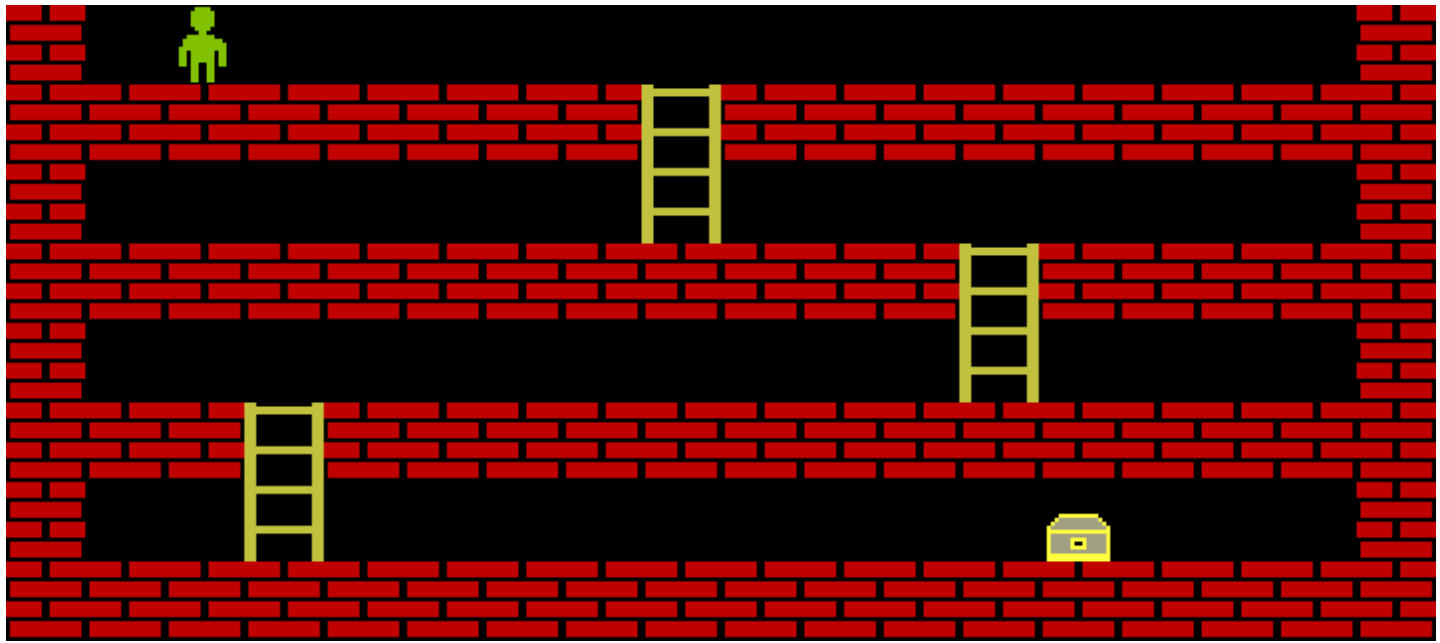
СОДЕРЖАНИЕ

3. ТРАЕКТОРИЯ ДВИЖЕНИЯ С ОБХОДОМ ПРЕПЯТСТВИЙ



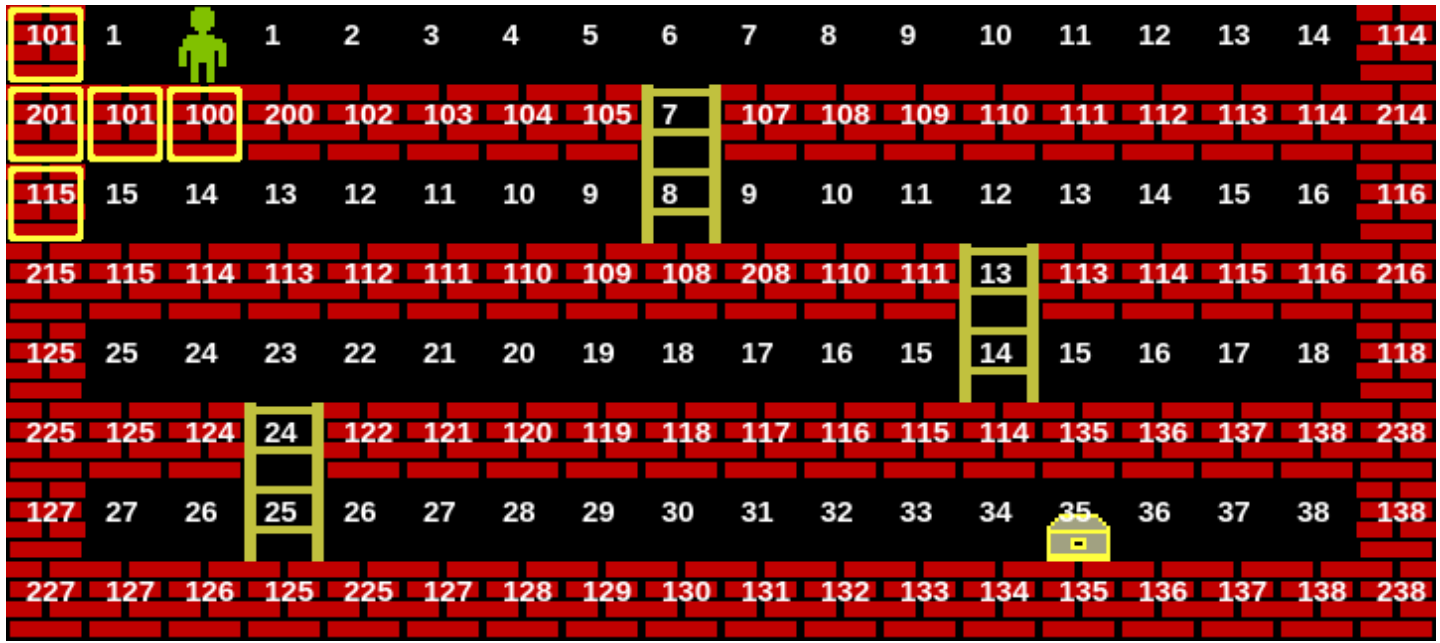
☐ Узлы ☑ Ребра ☐ Путь ☐ Стоимость Старт Стоп

3. ТРАЕКТОРИЯ ДВИЖЕНИЯ С ОБХОДОМ ПРЕПЯТСТВИЙ



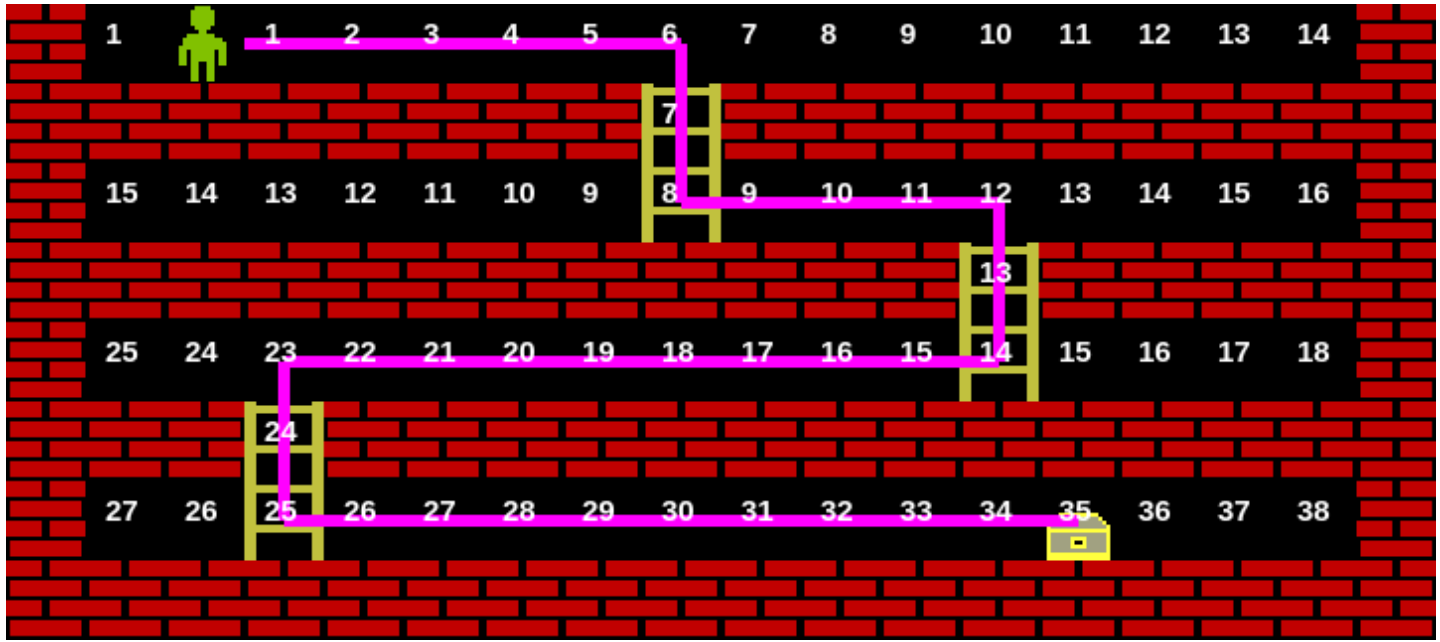
☐Узлы ☐Ребра ☐Путь ☐Стоимость Старт Стоп

3. ТРАЕКТОРИЯ ДВИЖЕНИЯ С ОБХОДОМ ПРЕПЯТСТВИЙ



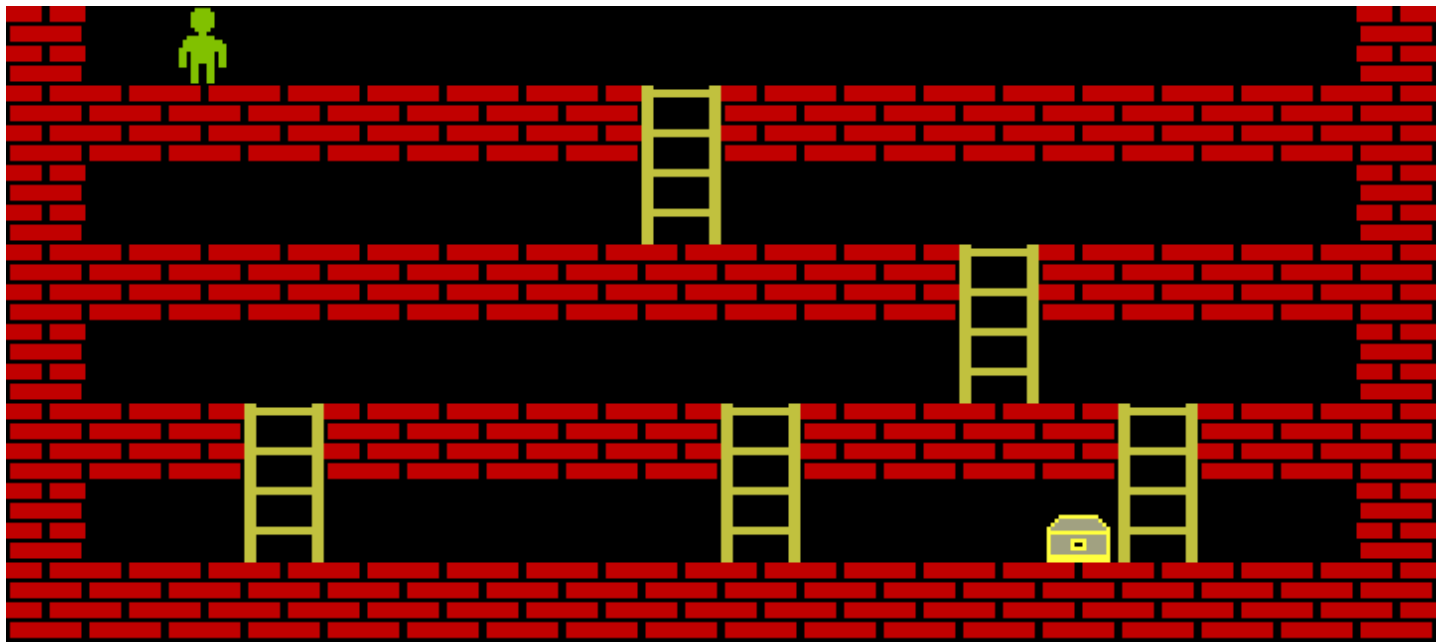
☐ Узлы ☐ Ребра ☐ Путь ☑ Стоимость Старт Стоп

3. ТРАЕКТОРИЯ ДВИЖЕНИЯ С ОБХОДОМ ПРЕПЯТСТВИЙ



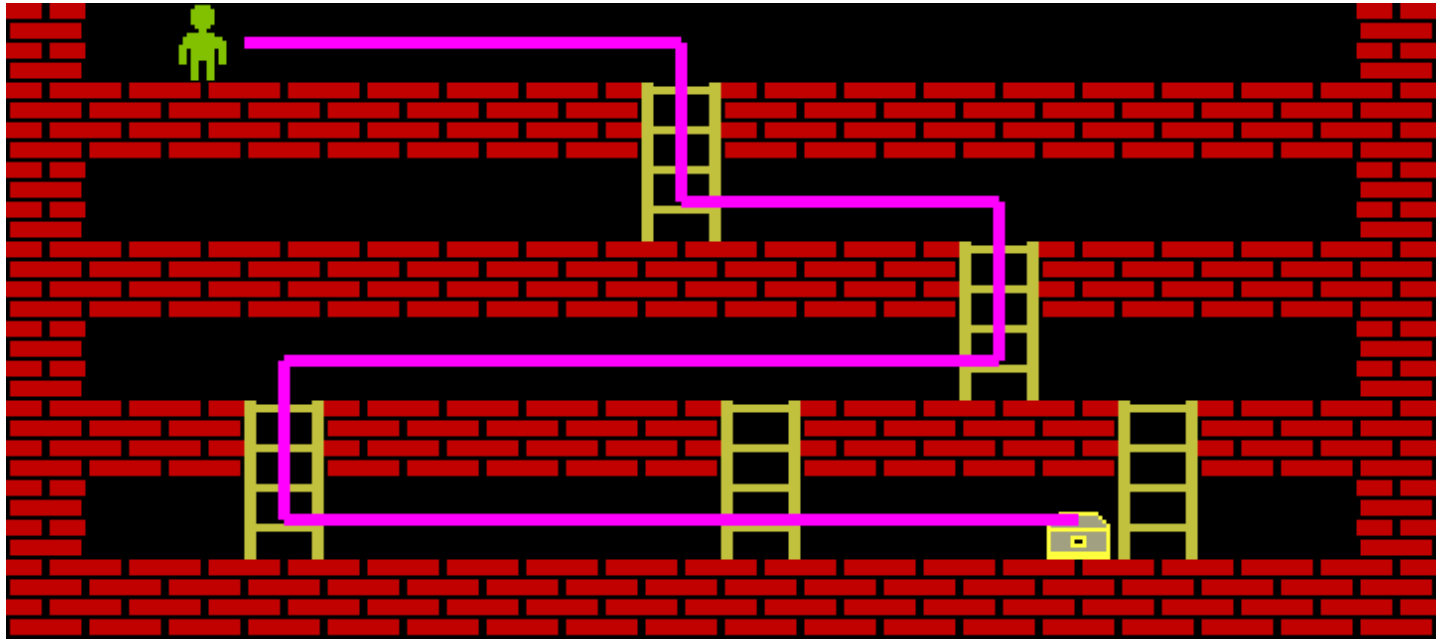
☐ Узлы ☐ Ребра ☑ Путь ☑ Стоимость Старт Стоп

3. ОШИБКА РЕАЛИЗАЦИИ: НЕ НАШЕЛ ОПТИМАЛЬНЫЙ ПУТЬ



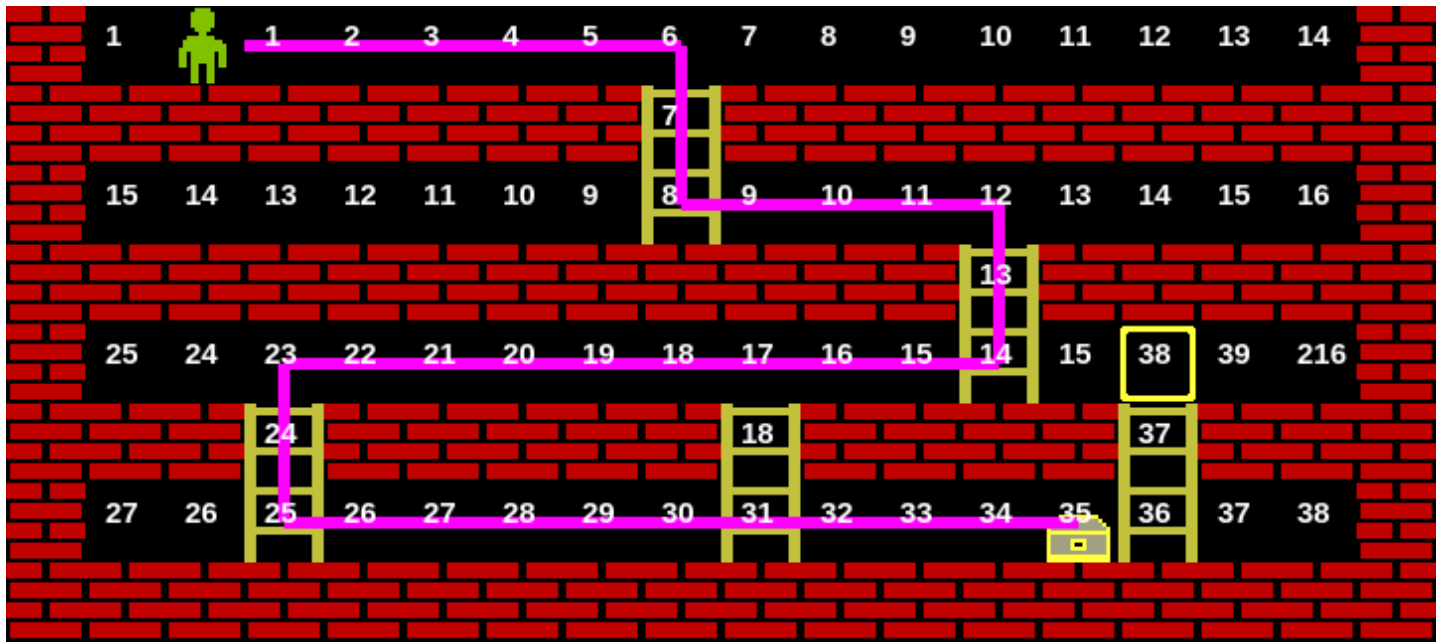
☐Узлы ☐Ребра ☐Путь ☐Стоимость Старт Стоп

3. ОШИБКА РЕАЛИЗАЦИИ: НЕ НАШЕЛ ОПТИМАЛЬНЫЙ ПУТЬ



☐ Узлы ☐ Ребра ☑ Путь ☐ Стоимость Старт Стоп

3. ОШИБКА РЕАЛИЗАЦИИ: НЕ НАШЕЛ ОПТИМАЛЬНЫЙ ПУТЬ

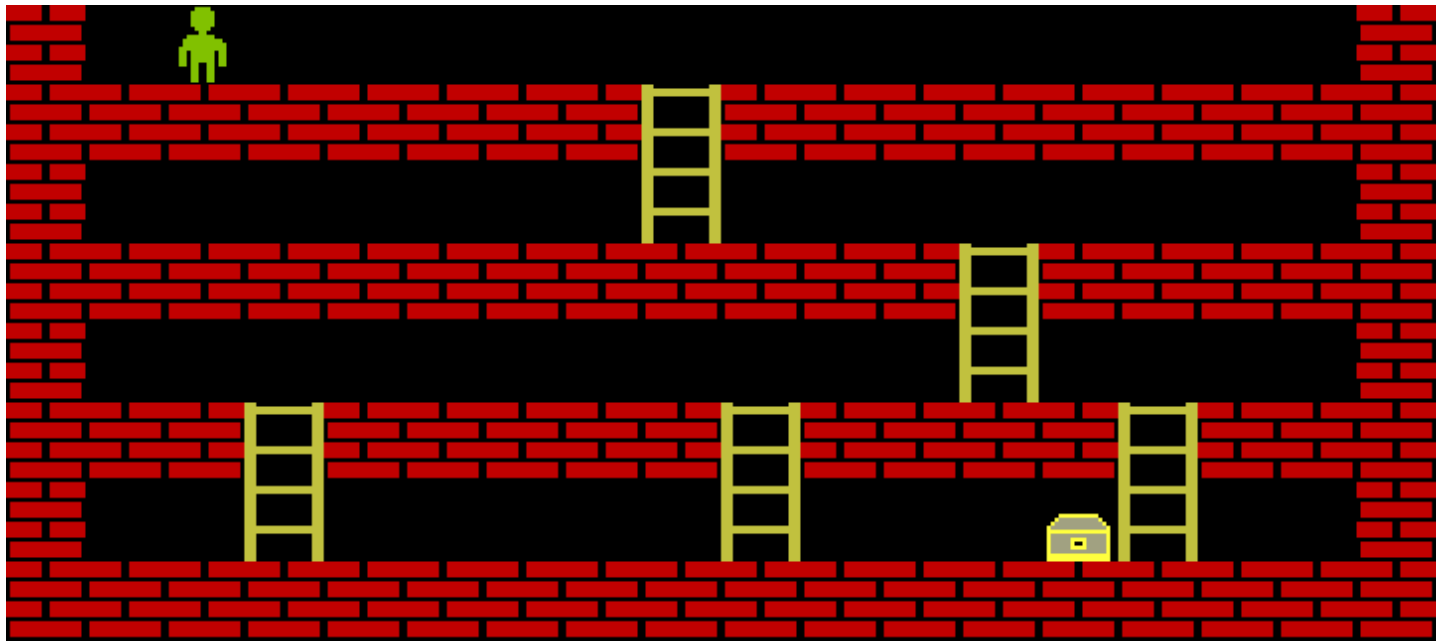


☐ Узлы ☐ Ребра ☑ Путь ☑ Стоимость Старт Стоп

3. ИСПРАВЛЕННАЯ ФУНКЦИЯ ВЫЧИСЛЕНИЯ СЛЕДУЮЩЕЙ ВЕРШИНЫ

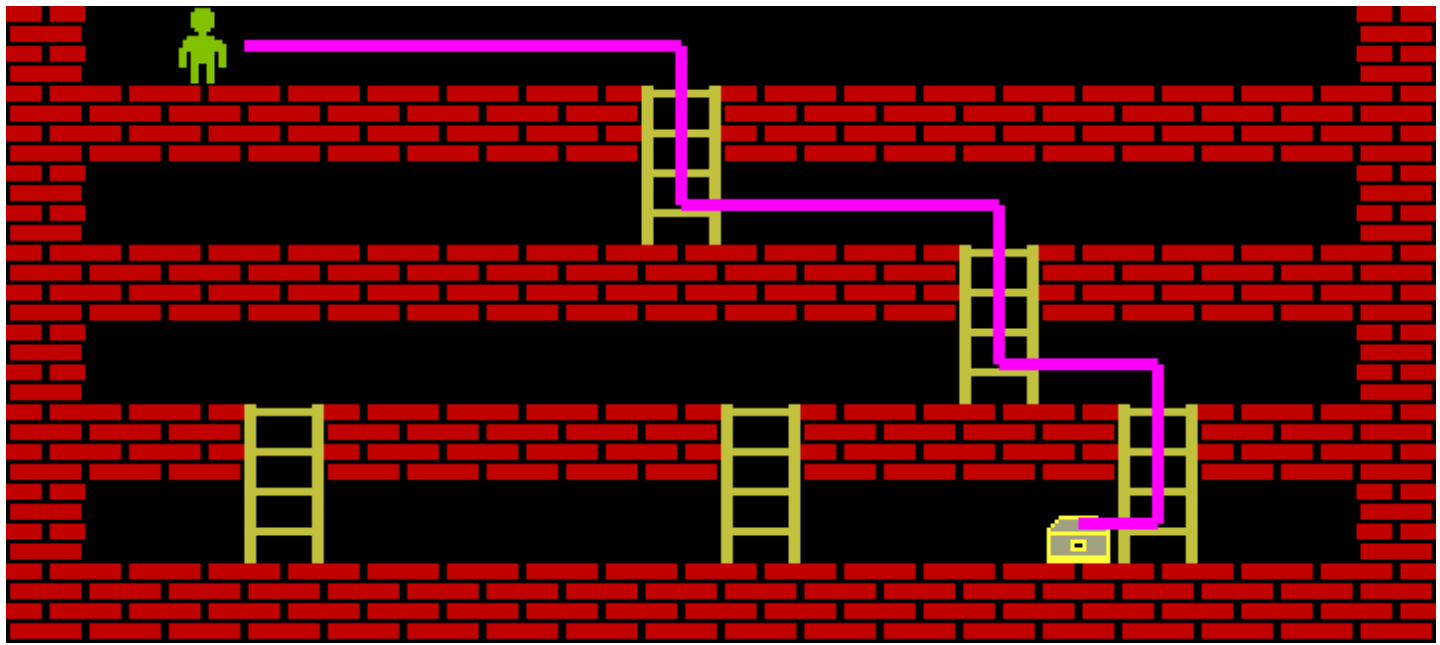

```
01. getNextVertex = (verticesToProcess: number[]): number => {
02.     let minAccessCost = Number.MAX_SAFE_INTEGER;
03.     let result = -1;
04.     verticesToProcess.forEach((nodeIndex) => {
05.         const vertex = this.vertices[nodeIndex as number];
06.         if (vertex.processed === false &&
07.             vertex.accessCost < minAccessCost)
08.         {
09.             minAccessCost = vertex.accessCost;
10.             result = nodeIndex as number;
11.         }
12.     });
13.     return result;
```

3. РЕЗУЛЬТАТ ВЫБОРА ОПТИМАЛЬНОГО ПУТИ



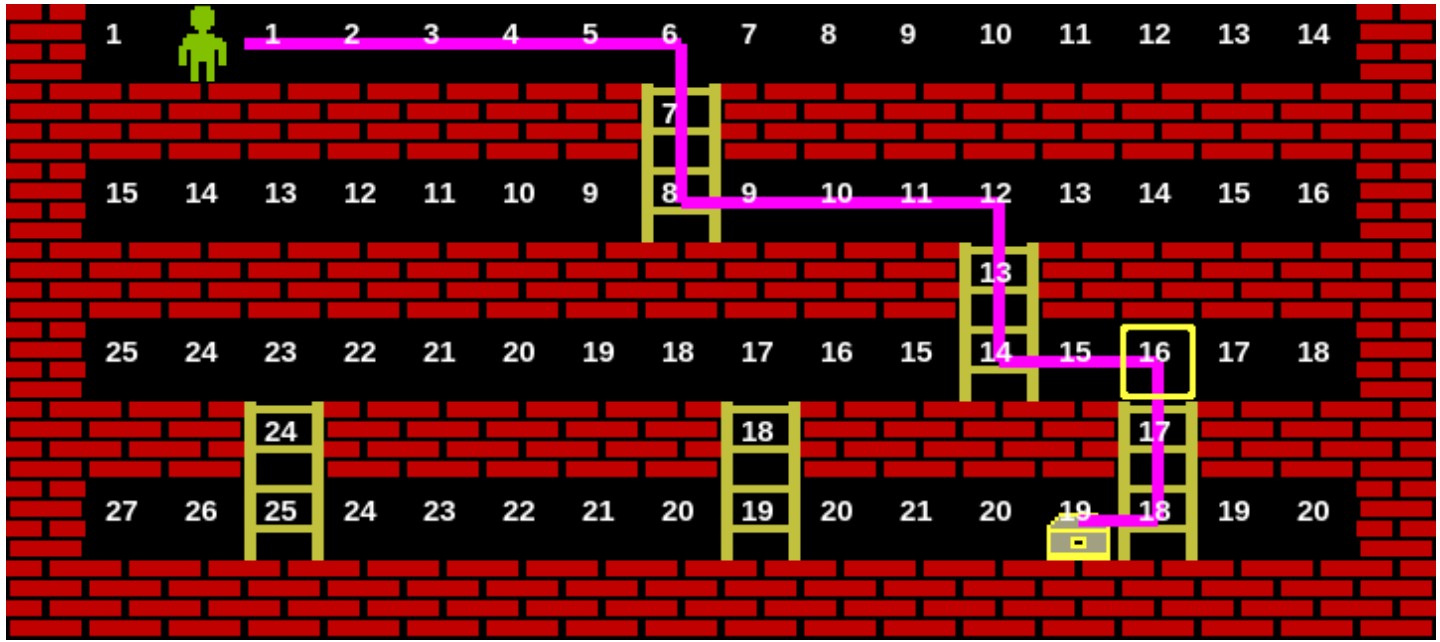
☐ Узлы ☐ Ребра ☐ Путь ☐ Стоимость Старт Стоп

3. РЕЗУЛЬТАТ ВЫБОРА ОПТИМАЛЬНОГО ПУТИ



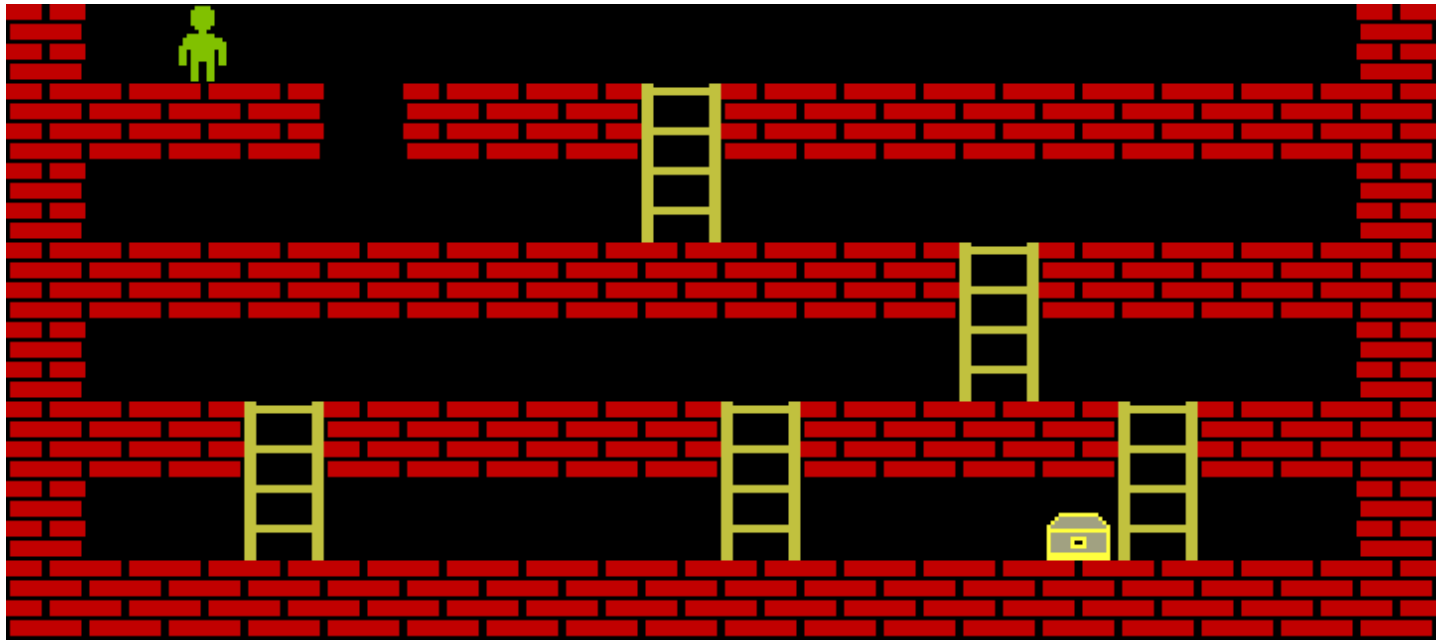
☐ Узлы ☐ Ребра ☑ Путь ☐ Стоимость Старт Стоп

3. РЕЗУЛЬТАТ ВЫБОРА ОПТИМАЛЬНОГО ПУТИ



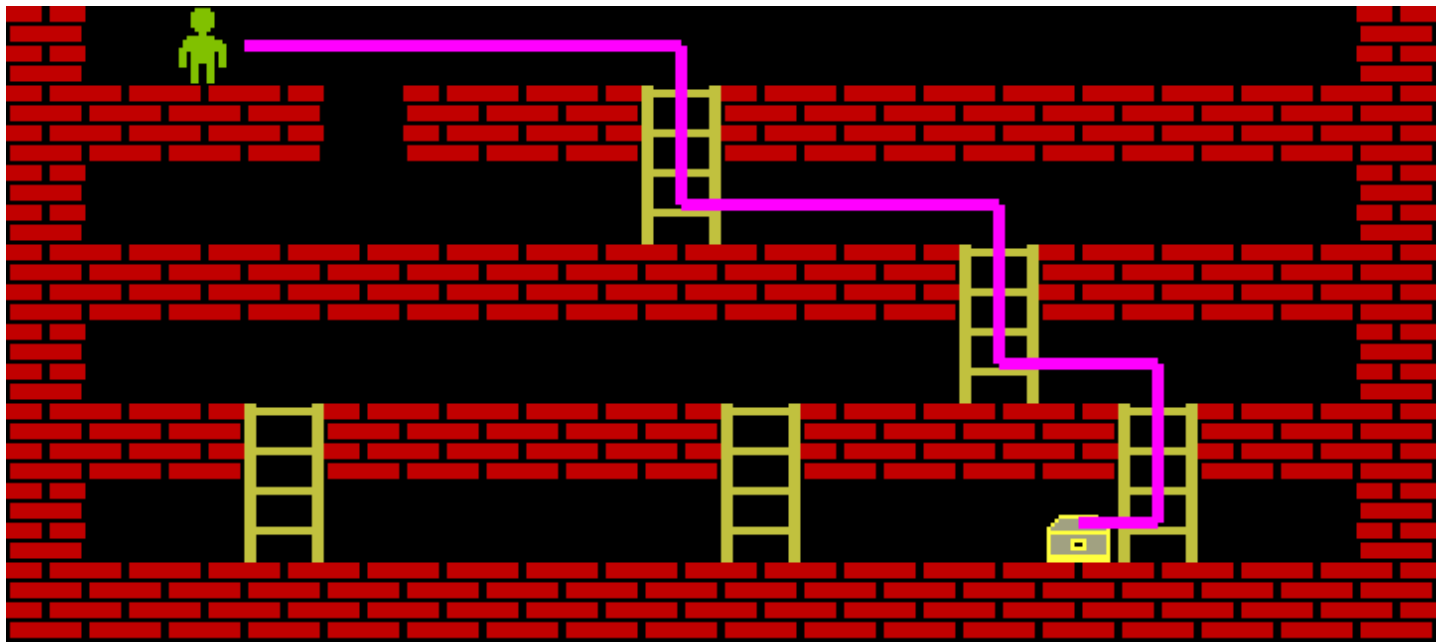
☐ Узлы ☐ Ребра ☑ Путь ☑ Стоимость Старт Стоп

3. ПЕРСОНАЖ ДОЛЖЕН ПАДАТЬ В ОТВЕРСТИЕ В ПОЛУ



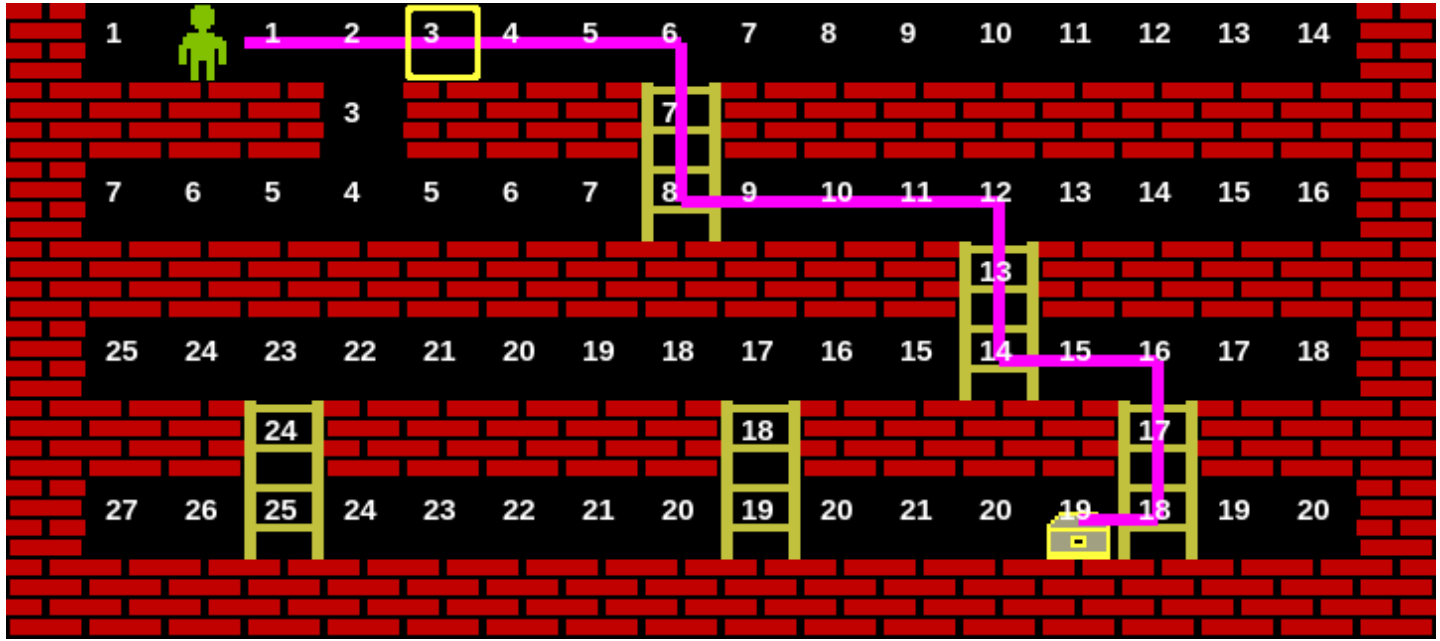
☐ Узлы ☐ Ребра ☐ Путь ☐ Стоимость Старт Стоп

3. ПЕРСОНАЖ ДОЛЖЕН ПАДАТЬ В ОТВЕРСТИЕ В ПОЛУ



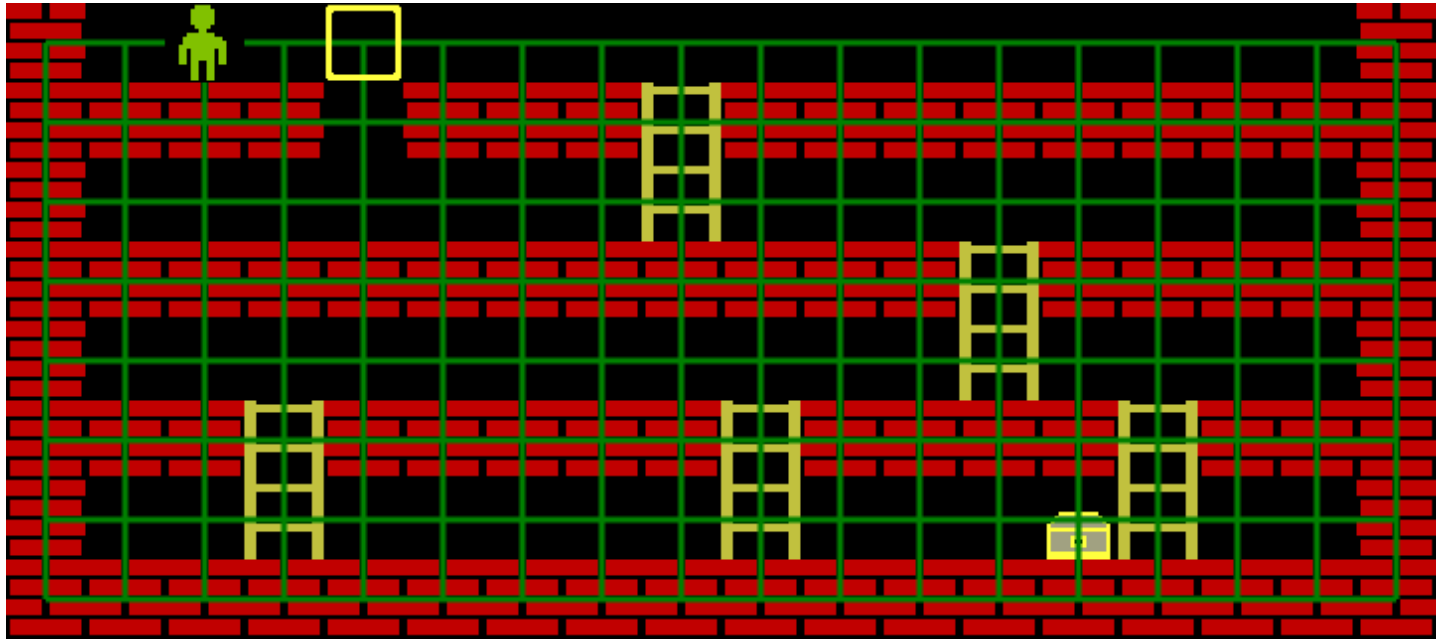
☐ Узлы ☐ Ребра ☑ Путь ☐ Стоимость Старт Стоп

3. ПЕРСОНАЖ ДОЛЖЕН ПАДАТЬ В ОТВЕРСТИЕ В ПОЛУ



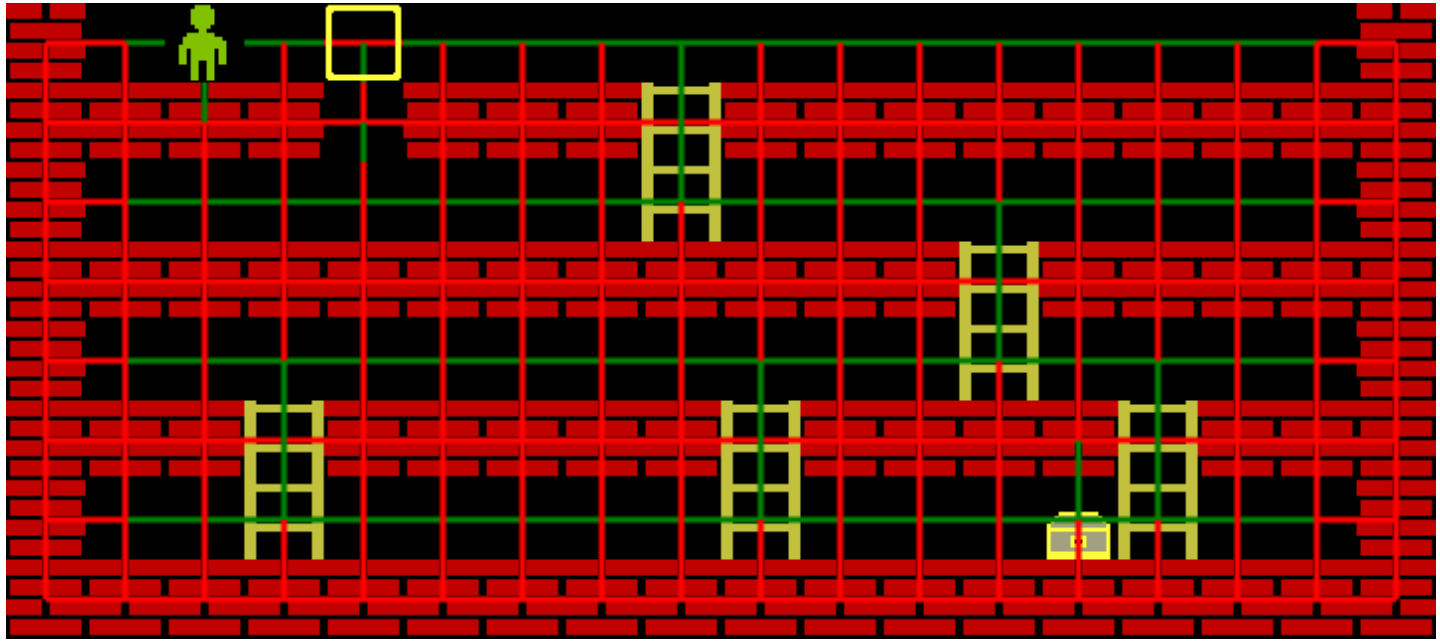
☐ Узлы ☐ Ребра ☑ Путь ☑ Стоимость Старт Стоп

3. ПЕРСОНАЖ ДОЛЖЕН ПАДАТЬ В ОТВЕРСТИЕ В ПОЛУ



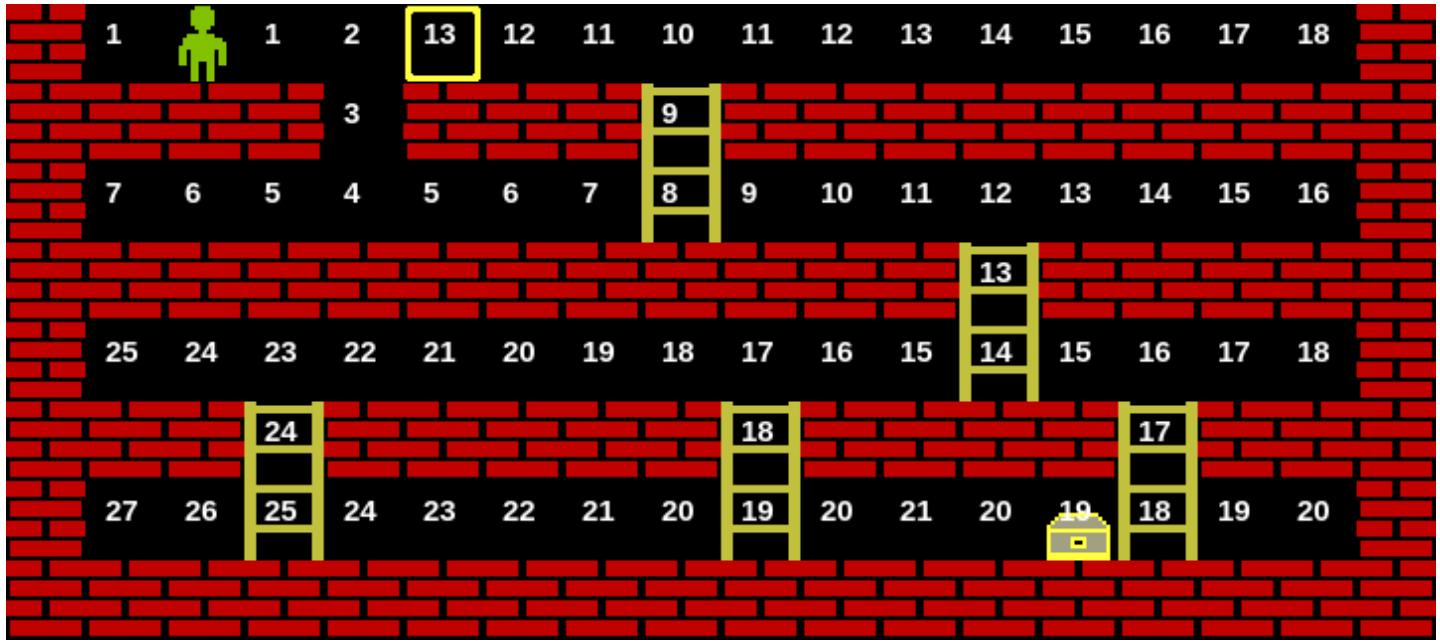
☐ Узлы ☑ Ребра ☐ Путь ☐ Стоимость Старт Стоп

3. ПЕРСОНАЖ ДОЛЖЕН ПАДАТЬ В ОТВЕРСТИЕ В ПОЛУ



☐ Узлы ☑ Ребра ☐ Путь ☐ Стоимость Старт Стоп

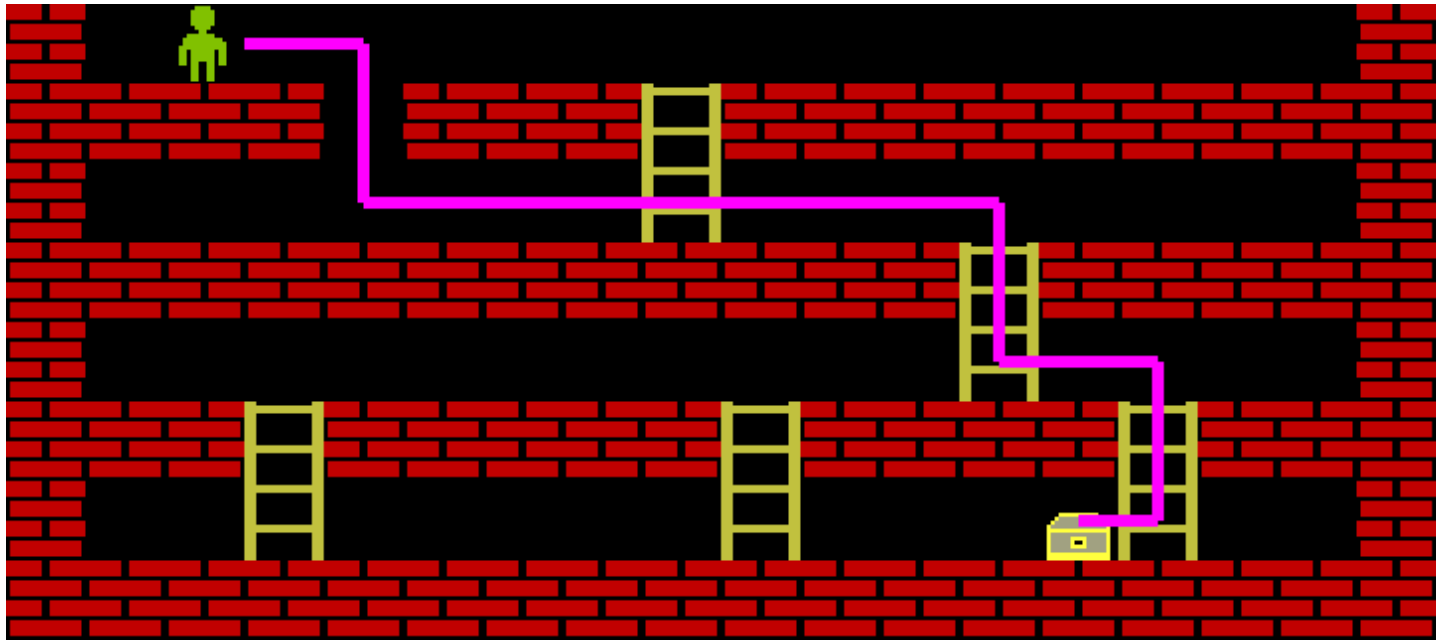
3. ПЕРСОНАЖ ДОЛЖЕН ПАДАТЬ В ОТВЕРСТИЕ В ПОЛУ



☐ Узлы ☐ Ребра ☐ Путь ☑ Стоимость Старт Стоп

3. ПЕРСОНАЖ ДОЛЖЕН ПАДАТЬ В ОТВЕРСТИЕ В ПОЛУ

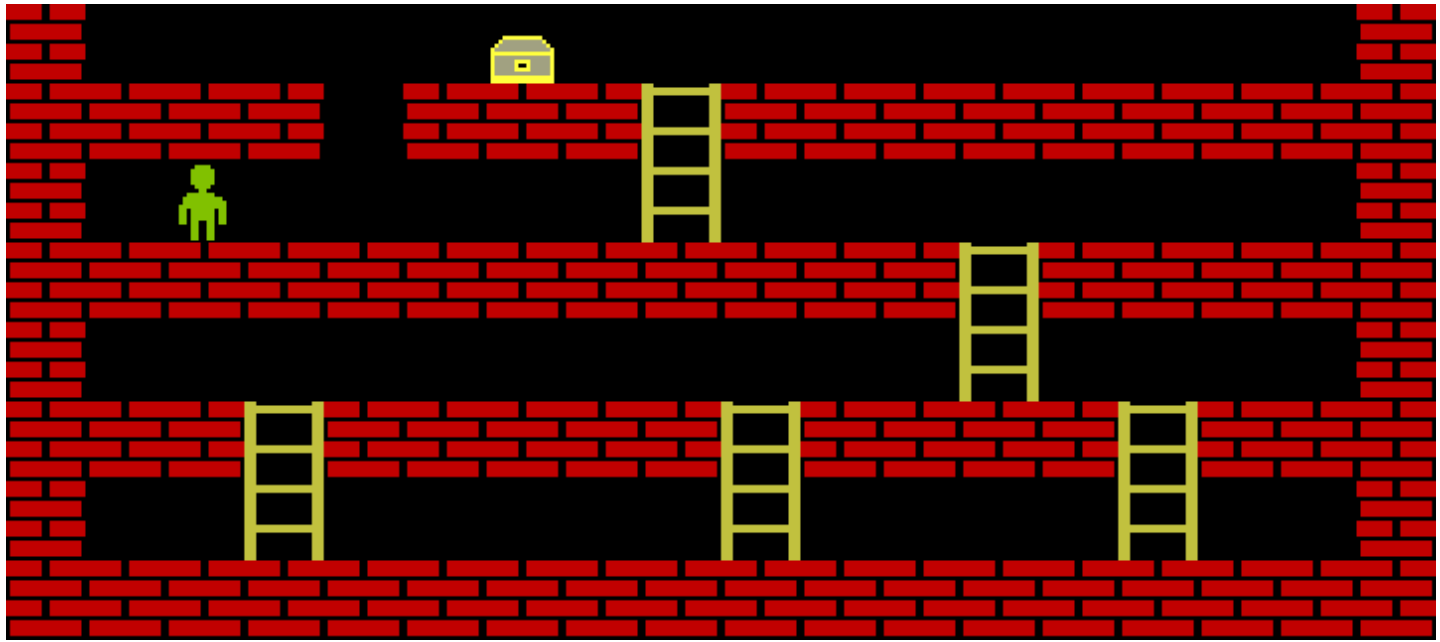
3. ПЕРСОНАЖ ДОЛЖЕН ПАДАТЬ В ОТВЕРСТИЕ В ПОЛУ



☐ Узлы ☐ Ребра ☑ Путь ☐ Стоимость Старт Стоп

3. ПЕРСОНАЖ НЕ ДОЛЖЕН ПОДНИМАТЬСЯ ВВЕРХ

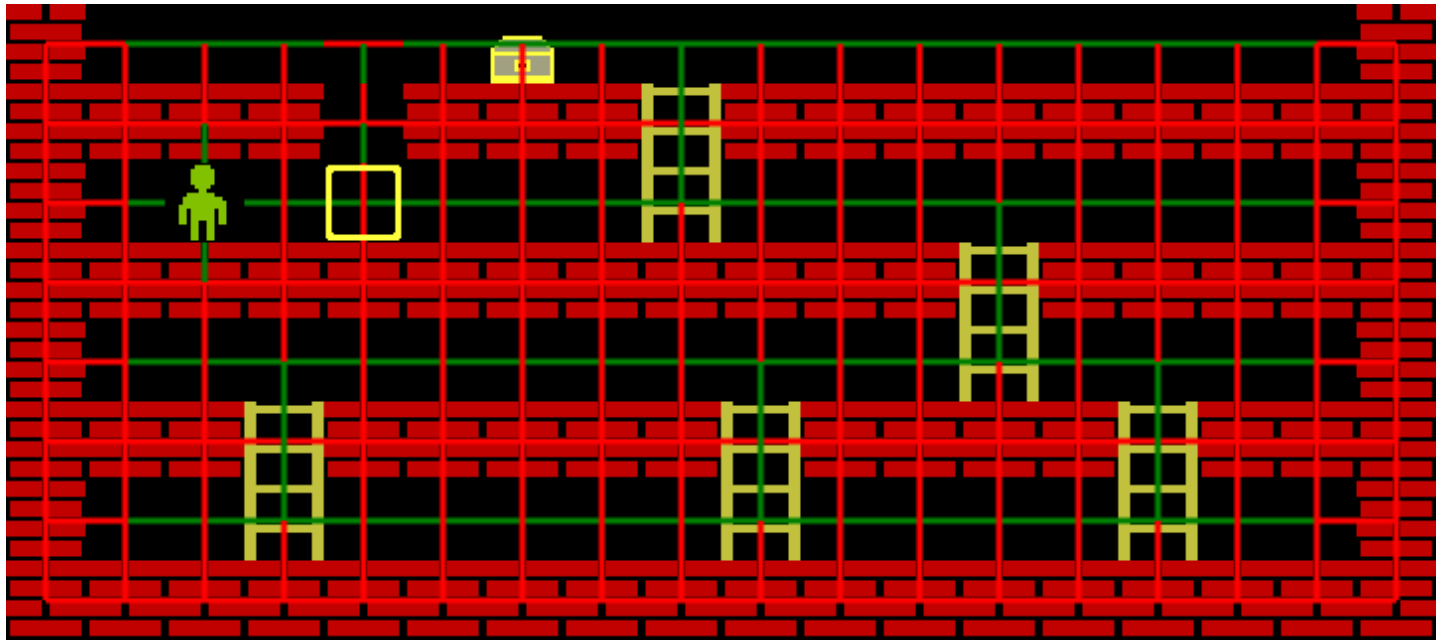
3. ПЕРСОНАЖ НЕ ДОЛЖЕН ПОДНИМАТЬСЯ ВВЕРХ ПО ВОЗДУХУ



☐Узлы ☐Ребра ☐Путь ☐Стоимость Старт Стоп

3. ПЕРСОНАЖ НЕ ДОЛЖЕН ПОДНИМАТЬСЯ ВВЕРХ

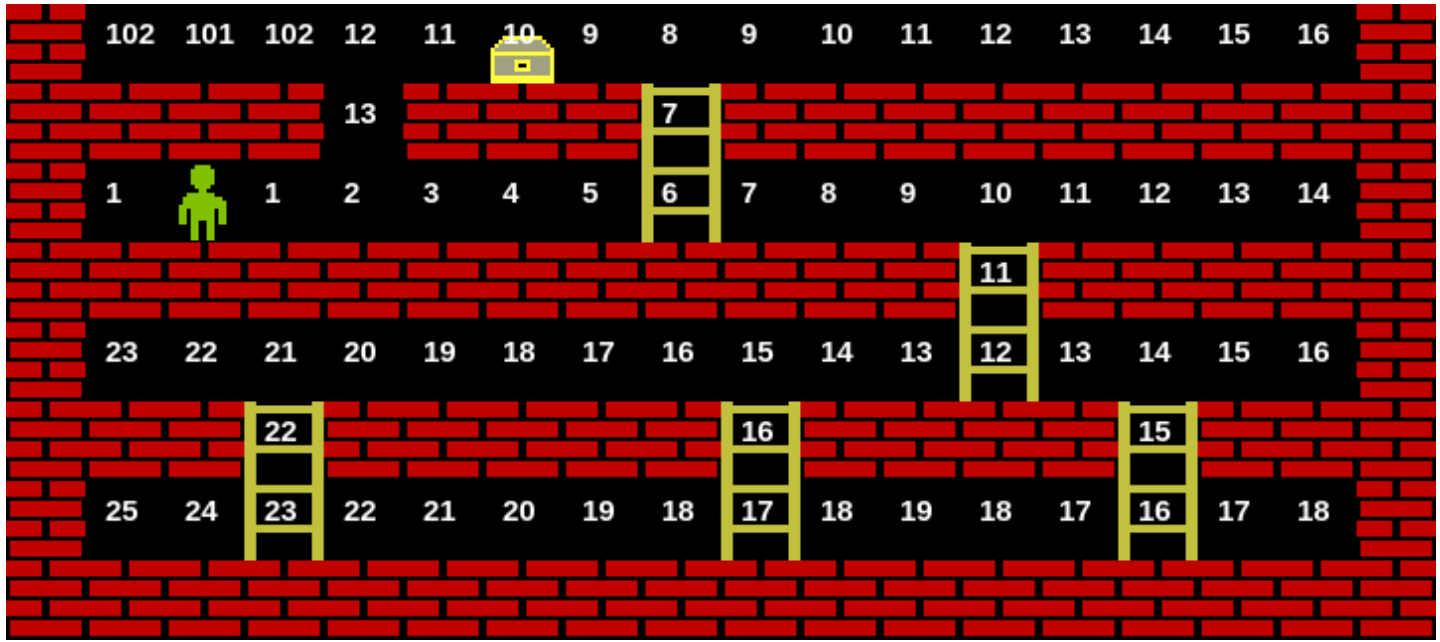
3. ПЕРСОНАЖ НЕ ДОЛЖЕН ПОДНИМАТЬСЯ ВВЕРХ ПО ВОЗДУХУ



☐ Узлы ☑ Ребра ☐ Путь ☐ Стоимость Старт Стоп

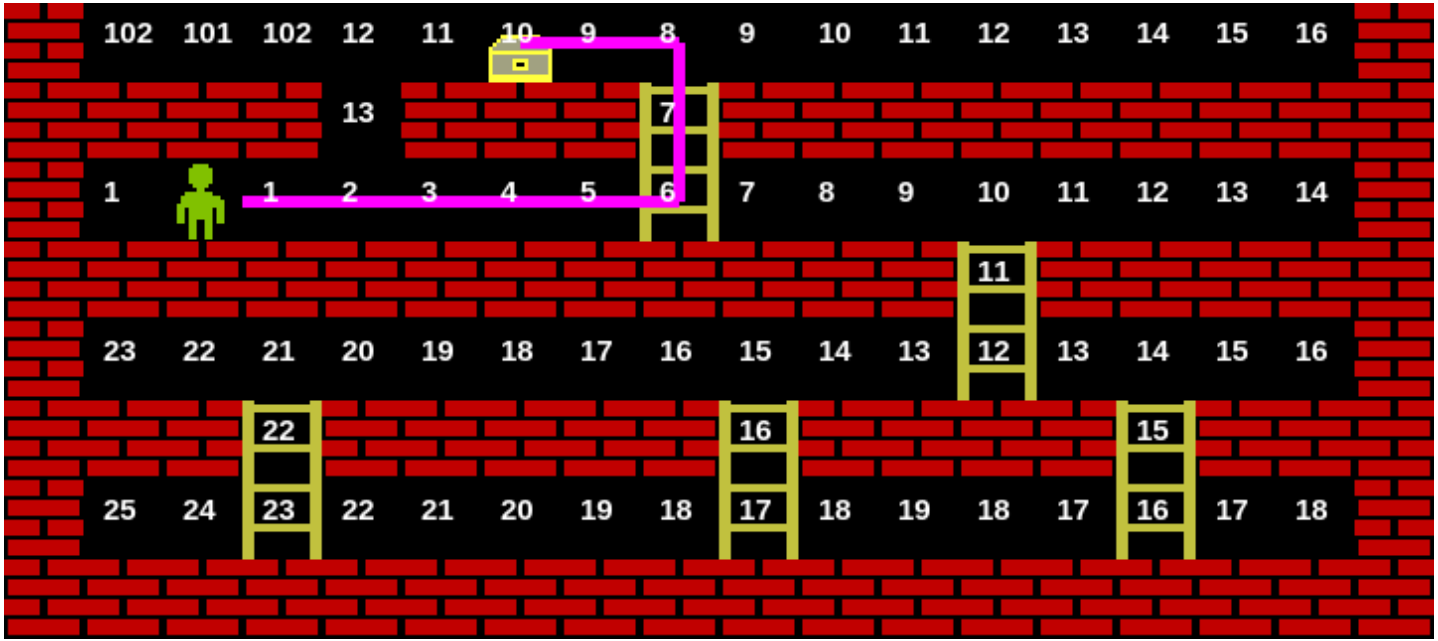
3. ПЕРСОНАЖ НЕ ДОЛЖЕН ПОДНИМАТЬСЯ ВВЕРХ

ПО ВОЗДУХУ



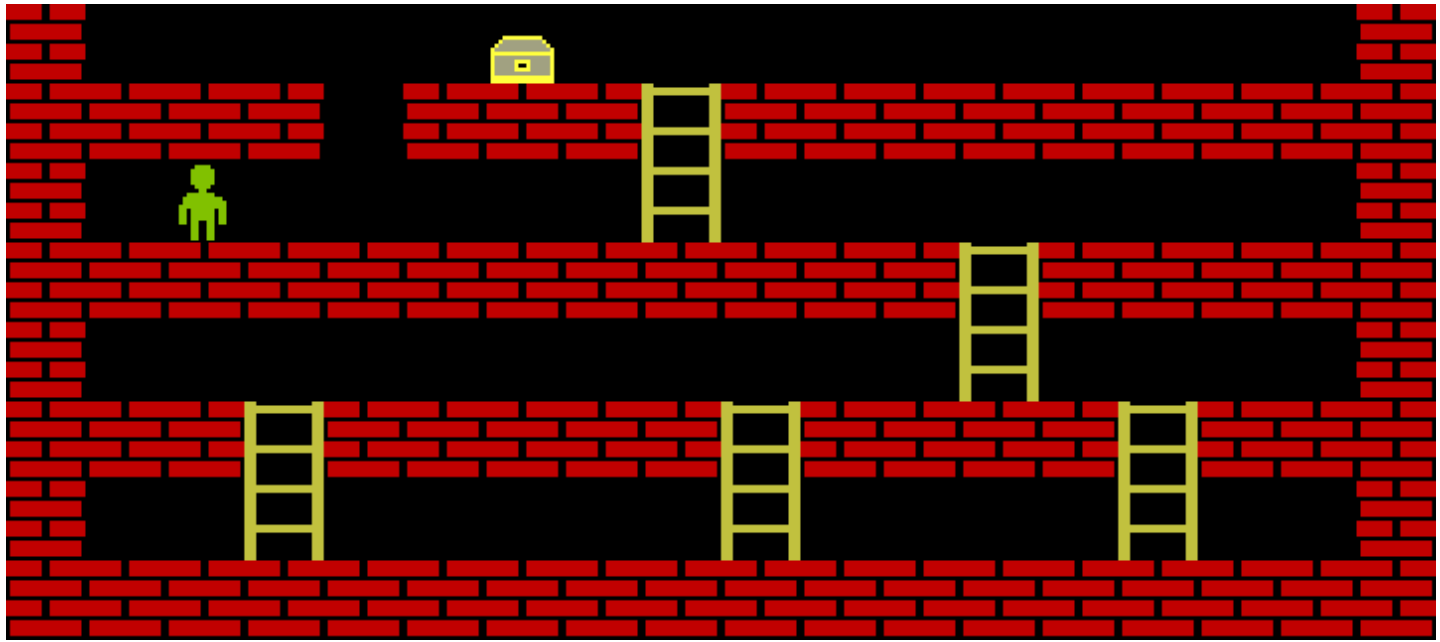
☐Узлы ☐Ребра ☐Путь ☑Стоимость Старт Стоп

3. ПЕРСОНАЖ НЕ ДОЛЖЕН ПОДНИМАТЬСЯ ВВЕРХ



☐ Узлы ☐ Ребра ☑ Путь ☑ Стоимость Старт Стоп

3. РЕЗУЛЬТАТ



☐Узлы ☐Ребра ☐Путь ☐Стоимость Старт Стоп

СОДЕРЖАНИЕ

1. Алгоритм Дейкстры

2. Адаптация алгоритма Дейкстры к работе на графе - квадратной сетке

3. Решение задач расчета траектории движения

ВЫВОДЫ

1. Фронтенд - это еще и про алгоритмы

2. Можно создавать свои игры на основе работающего примера:
https://github.com/alexanderpono/bricks-runner

3. Не бойтесь экспериментировать. Преимущество фронтенда – можно сразу увидеть результат!

ЧТО ЕЩЕ ПОЧИТАТЬ?

- A* - развитие алгоритма Дейкстры
  https://www.redblobgames.com/pathfinding/a-star/introduction.html
- Еще об алгоритмах на графах
  https://habr.com/ru/companies/timeweb/articles/751762/

СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ?

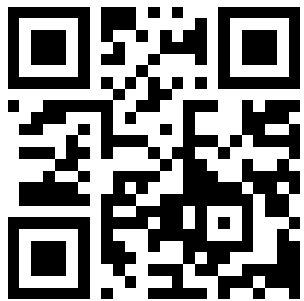
Александр Пономаренко

Описание A*

Алгоритмы на графах

Исходники игры

Александр Пономаренко



Новости ИТ в Иннотехе и Т1