[**Oleksandr Roshchupkin**](https://app.slack.com/team/UC3GARWKX)  [1:31 PM](https://fs-danit.slack.com/archives/C02EDJ1CNDA/p1642937463004400)

Важный момент по поводу последней задачи - координаты декартовой системы не будут совпадать с индексами массива!  
Поэтому при обращении к массиву нодов, индексы надо пересчитывать

**Hamiltonian path problem**.

Когда змейка значительно удлинняется- включать алгоритм Хамильтона, суть которого –в том, чтобы не ходить одним и тем же путём дважды –это помогает змейке быть компактной.

Если сделать правильный Хамильтон и срезаться в районе 80ти, то можно сделать результат 2х AI

Нужно глянуть на алгоритм волнового распределения Ли – из простых – самый действенный для змейки. -Это очень крутой алгоритм: бросаем камушек в воду и смотрим, где распространяются волны (у препятствий волны исчезнут). Самый действенный из простых алгоритмов: идем на зеленое, если не можем добраться – идем на красное, если и до красного уже не можем добраться – кусаем хвост. Если не можем дойти к зеленому- мы либо слишком длинные, либо херовый алгоритм (либо совсем дурная ситуация).

Очень тщательно следить, чтобы не перепутать X и Y

Когда нашли цель, нужно сразу просчитать backTrace , чтобы просчитать как змею компактно уложить на обратном ходе. Змею нужно структурировать в LinkedList –е

Наша задача – написать всего одну-единственную функцию, которая принимает Board , а выдаёт Direction.

Перерасчет графа выполнять на каждом шаге, т.к. хвост змеи –это динамически меняющееся и препятствие возможное открытие более оптимального пути.

Продолжить изучать видео рыхальского с 1:16:00

JavaAlgorythm9 2 Snake game introduction 21 03 2021.mp4

D:\Video\_Dan\_IT\! Java\А Рыхальский\Java\_3 Алгоритмы