АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ЛОДКИ

Да започнем да правим наблюдения върху задачата. Задачата ни се свежда до това да поддържаме изпъкналата обвивка на множеството от точки и да следим кога някоя точка напуска изпъкналата обвивка.

Една (грешна) идея за решение би била да вземем позициите на точките в момент далеч в бъдещето, да изградим изпъкналата обвивка на точките и да сравним с първоначалната изпъкнала обвивка. Ако няма изменение, тогава ще търсим изяждане на лодка в по-късен момент от време. Ако има промяна, ще търсим промяна в по-ранен момент. Изглежда все едно можем да правим двоично търсене, но всъщност това не е вярно решение на задачата. Възможно е една точка да излезе от изпъкналата обвивка и да се появи на друго място. Освен това е възможно точките да излязат и да влязат в обвивката и пак да застанат в същата ориентация, както са били преди. Решение, което опитва двоично търсене по времето на следващата промяна и строи изпъкнала обвивка на всяка стъпка ще хване комбинация от WA, OK и TL резултати на тестовете и около 30-40 точки.

Възможно е да правим симулация на процеса като взимаме състояния с нарастващо време и следим за промяна. Ако имаме промяна между две стъпки, можем да намалим стъпката време и да търсим точния момент на промяна с по-висока точност. На всяка стъпка, за да открием дали сме имали промяна спрямо предходния момент от време можем да строим изпъкнала обвивка на точките, или да проверим ориентацията на всички точки, дадена в реда от предната изпъкнала обвивка. Ако при движение по точките в ред, дефиниран от предната изпъкнала обвивка (в ред обратен на часовниковата стрелка), правим само леви завои, тогава изпъкналата обвивка не се е променила. Това какви завои правим може да бъде сметнато с помощта на векторно произведение и ориентирани лица (https://www.topcoder.com/community/data-science/data-science-tutorials/geometry-concepts-basic-concepts/#cross_product). Подобна техника с използване на ориентирани лица се ползва от алгоритъма на Graham за изпъкнала обвивка. Решение, което внимателно взима нарастващи стъпки и не строи изпъкнала обвивка на всяка стъпка ще получи около 50-60 точки.

Едно важно наблюдение върху изпъкнали многоъгълници е, че ъгълът, дефиниран от три съседни точки върху изпъкналата обвивка, гледащ към вътрешността на многоъгълника е по-малък от 180 градуса. И още повече, ако при движение на точките всички ъгли останат по-малки от 180 градуса във всеки един момент, тогава никоя точка няма да напусне изпъкналата обвивка. Ако можем да идентифицираме моментите когато ъглите между три съседни върха стават равни на 180 градуса, това ще са точно моментите, в които лодки биват изядени – точно тогава те спират да бъдат върхове на многоъгълника. Това може да бъде направено чрез ориентирани лица. Знакът на ориентираното лице ни дава посоката на въртене, за да отидем от един вектор в друг. Ако имаме три последователни точки върху изпъкналата обвивка в ред обратен на часовниковата стрелка, A, B, C, тогава векторното произведение (A-B) x (C – B) ще има стойност по-малка от нула. (В действителност векторното произведение е 3d вектор. Тук използваме само големината на векторното произведение по посока положителната z ос). В момента, когато лодка B бива изядена, векторното произведение ще бъде равно на нула (момента, в който то си сменя знака).

Формулата за големина на векторно произведение между вектори:

A-B=(x1, y1) и C-B=(x2, y2) е x1*y2 – x2*y1. Неравенството P(t) = x1*y2 – x2*y1 < 0 ще се наруши когато P(t) = 0. Формулата зависи от t – време, защото координатите на точките се изменят с времето. След като си разпишем формулата, ще получим квадратно уравнение за t, и трябва да намерим най-малката положителна стойност на t, за която P(t) = 0. Така ще намерим първия момент, когато средната точка B напуска изпъкналата обвивка. Решаването на квадратно уравнение е възможно чрез използване на формула с константна

сложност.

Това вече е достатъчно, за да напишем решение. Необходимо е да следим N пресечни точки, и да актуализираме изпъкналата обвивка, когато някоя точка напусне обвивката. Можем да поддържаме изпъкналата обвивка като цикличен двойно свързан списък (всяка точка знае номерата на двете си съседни точки). Така можем лесно да премахваме точка от изпъкналата обвивка – необходимо е само да актуализираме двата съседа. Можем да държим моментите на напускане на изпъкналата обвивка, заедно с участващите точки в приоритетна опашка, подредена по време (най-малкия момент от време е на върха). За всяко изяждане на лодка B трябва да пазим четворка (t, A, B, C), където A, B, C са трите съседни точки. Ако някоя от точките A, B, C вече не присъства в изпъкналата обвивка, нашата четворка е невалидна към сегашния момент (някоя друга от точките A или C е напуснала изпъкналата обвивка по-рано). Общо, всяка точка ще напусна обвивката най-много веднъж. Когато една точка напусне опашката, трябва да добавим до две нови събития в приоритетната опашка – моментите на напускане на изпъкналата обвивка, дефинирани от точки които тепърва стават съседи (защото сме премахнали точка, нови двойки точки стават съседи върху изпъкналата обвивка). Тоест, цялата сложност на алгоритъма е O(n * log(n)).

Автор: Йордан Чапъров