الف) در این روش، ما فاصله بین هر دو نقطه ممکن را حساب می‌کنیم. انتخاب هر جفت نقطه ممکن $n(n-۱)/۲$ حالت مختلف می‌شود که از $O(n^۲)$ است. این الگوریتم برای n های بزرگ کند است و ما به الگوریتم دیگری نیاز داریم. برای جواب مسأله نیازی به محاسبه فاصله تمام جفت نقاط نیست. برای بهتر کردن الگوریتم، ایده کلی به این صورت است که تعداد جفت نقاطی که فاصله‌شان را حساب می‌کنیم را کاهش دهیم.

ب) برای کاهش دادن مقایسه‌های غیرضروری، از رویکرد تقسیم و حل استفاده می‌کنیم. ابتدا نقاط را بر اساس مختصات x شان مرتب می‌کنیم که این کار در $O(nlogn)$ انجام می‌شود. سپس عناصر الگوریتم تقسیم و حل به این صورت می‌شود:

۱. تقسیم: نقاط را بر حسب x شان از وسط، به دو نیمه چپ و راست تقسیم می‌کنیم.

۲. بازگشت: به صورت بازگشتی نزدیک‌ترین نقاط را نیمه چپ و راست پیدا می‌کنیم.

۳. ترکیب: نزدیک‌ترین جفت نقطه برای هر مرحله، علاوه بر نزدیک‌ترین جفت نقطه در نیمه چپ و راست که به صورت بازگشتی پیدایشان کردیم، ممکن است در امتداد و نزدیکی خط تقسیم ما قرار داشته باشد. بنابراین، باید نقاطی که در یک نوار باریک با عرض $۲d$ قرار دارند را بررسی کنیم، که در آن d کوچک‌ترین فاصله بین دو نیمه است. این نقاط را بر اساس مختصات y شان مرتب کرده و فقط همسایگان نزدیک را بررسی می‌کنیم.

نکته هوشمندانه در همین قسمت ترکیب است. ما به جای مقایسه تمام جفت نقاط در نوار، هر نقطه را حداکثر باید با ۶ نقطه دیگر در نوار بررسی کنیم که این کار تعداد مقایسه‌های غیرضروری را کاهش می‌دهد. بدیهی است که اگر دو نقطه فاصله افقی‌شان بیشتر از d باشد قبلاً حذف شده‌اند. پس اگر یک نقطه را در نظر بگیریم، فقط باید نقاطی که فاصله عمودی آن‌ها کمتر از d باشد را بررسی کنیم. اگر بیشتر از ۷ نقطه وجود داشته باشد، حداقل دو نقطه وجود خواهند داشت که در فاصله کمتر از d از یکدیگر قرار می‌گیرند. به صورت شهودی می‌توانیم مستطیلی با طول $۲d$ و عرض d را فرض کنیم که حداکثر ۶ نقطه با شرایط ما در آن قرار می‌گیرند.

پ) با $O(۱)$ در قسمت تقسیم، مسئله را به دو زیرمسئله از اندازه $n/۲$ تقسیم می‌کنیم. بررسی نوار در قسمت ترکیب نیز با $O(n)$ انجام می‌شود. پس رابطه بازگشتی را می‌توان به این صورت نوشت:

$$T(n) = ۲T(n/۲) + O(n)$$

برای حل رابطه بازگشتی از قضیه اصلی استفاده می‌کنیم.

$$a=۲, b=۲, f(n) = O(n)$$

طبق قضیه اصلی، چون $f(n) = O(n)$ و $O(n^{\log_2 2}) = O(n)$ است، این حالت دوم قضیه اصلی را نشان می‌دهد. پس پیچیدگی نهایی الگوریتم از $O(n\log n)$ می‌باشد که خیلی سریعتر از $O(n^2)$ است.

محمد کریمی ۴۰۲۳۶۱۳۰۶۰

امیررضا محمدی یگانه ۴۰۲۳۶۱۳۰۶۸