به نام خدا

علی غفاری ثابت - ۹۸۲۳۹۰۳

توضیح کد:

۱- تابع dist مختصات ۲ نقطه رو از ورودی به شکل لیست دریافت می کند و فاصله دو نقطه را بر می گرداند.

۲- تابع closest_dist به ترتیب لیستی از نقاط که بر اساس x مرتب شده اند، اندیس شروع و اندیس پایان را دریافت کرده و کوتاه ترین فاصله بین بازه داده شده از نقاط را بر می گرداند. این تابع به این شکل عمل می کند که اگر تعداد نقاط داده شده کوچکتر از ۳ بود کوتاه ترین فاصله را بین آنها به روش brute force بر می گردانیم. در غیر اینصورت بازه داده شده را از وسط به ۲ قسمت چپ و راست (اگر فرض کنیم نقطه وسط mp باشد آنگاه نقاط به دوسته سمت چپ و راست صفحه $x = mp_x$ تقسیم می شود.) تبدیل کرده و کوتاه ترین فاصله را در این دو زیر مجموعه پیدا می کنیم و مقدار کوچکتر را در متغیر c ذخیره می کنیم. حال باید فاصله نقاطی را که یکی در مجموعه چپ و دیگری در مجموعه راست قرار دارد را محاسبه کرده و اگر از c کوچکتر بود آن را جایگزین c کنیم. برای این کار به این روش عمل می کنیم که نقاطی از بازه داده شده را که فاصله آنها از خط $x = mp_x$, $z = mp_z$ کمتر از c است را در یک لیست ذخیره می کنیم و سپس آن لیست را بر اساس y نقاط مرتب می کنیم. حال به ازای هر نقطه در این لیست فاصله آن نقطه را با نقاط بعدی اش در لیست اندازه گیری کرده و اگر کمتر از c بود c را برابر آن مقدار قرار می دهیم. و این کار را برای هر نقطه تا جایی ادامه می دهیم که یا نقطه بعدی وجود نداشته باشد یا اختلاف y نقطه بعدی با y نقطه ای که در حال بررسی آن هستیم کمتر از c باشد. (چون لیست بر اساس y نقاط مرتب شده است اگر ادامه دهیم قطعا فاصله نقاط بعدی از نقطه ای که در حال بررسی آن هستیم بیشتر از c خواهد بود.) در نهایت c را به عنوان خروجی تابع بر می گردانیم.

۳- در نهایت عدد n را از ورودی خوانده و مختصات n نقطه را از کاربر دریافت می کنیم. سپس این n نقطه را بر اساس x نقاط مرتب می کنیم زیرا ورودی تابع closest_dist نقاط مرتب شده بر اساس x است. در نهایت مجموعه نقاط را به همراه اندیس شروع و پایان بازه ای که برای بررسی می خواهیم (در اینجا ۰ و n) را به تابع closest_dist داده و خروجی را به عنوان کمترین فاصله بین نقاط چاپ می کنیم.

پیچیدگی زمانی:

در بدنی اصلی برنامه یک بار تمام نقاط را بر اساس x مرتب می کنیم که پیچیدگی زمانی این بخش $\theta(nlg(n))$ است. در تابع closest نیز هر بار مسئله را به دور زیر مسئله با سایز $n/2$ تبدیل کرده و n عملیات برای محاسبه فاصله نقاط از خط $x = mp_x$, $z = mp_z$ و افزودن آن به لیست mid_points انجام می شود. همچنین پیچیدگی زمانی مرتب سازی این لیست برابر $\theta(nlg(n))$ است. در نهایت یک حلقه برای بررسی فاصله بین نقاط mid_points وجود دارد که به خاطر شرط اینکه اختلاف y نقاط بعدی نباید بیشتر از c باشد برای هر نقطه $\theta(1)$ عملیات نیاز داریم زیرا با اصل لانه کبوتری ثابت می شود که حداکثر ۱۴ نقطه می تواند وجود

داشته باشد که اختلاف y آن با نقطه ای که در حال بررسی آن هستیم کمتر از c باشد زیرا در غیر این صورت مقدار c باید کمتر می شد. در نتیجه پیچیدگی زمانی این تابع برابر است با:

$$T(n) = 2T(n/2) + \theta(nlg(n)) \simeq \sum_{0}^{lg(n)-1} 2^i \frac{n}{2^i} lg(\frac{n}{2^i}) = \sum_{0}^{lg(n)-1} nlg(n) - i = n * lg(n) * lg(n) - \sum_{0}^{lg(n)-1} i$$

$$= n * lg(n) * lg(n) - lg(n) * (lg(n) - 1)/2 = \theta(n * lg(n) * lg(n))$$

و با توجه به اینکه پیچیدگی زمانی بدنه اصلی برنامه برابر $\theta(nlg(n))$ است می توانیم نتیجه بگیریم پیچیدگی زمانی کل برنامه برابر $\theta(n * lg(n) * lg(n))$ است.

تصویر محیط اجرا: