E- Connected Treasure Box

原案・解説 tardigrade(akTARDIGRADE13)

2023 年 5 月 4 日

集合 S に含まれる点を頂点とし、マンハッタン距離が d 以下の頂点同士には辺を張ることにします。「最終的に S に含まれる頂点同士のマンハッタン距離は $\frac{d}{2}$ 以上」という制約から、辺の本数は陽に管理できる程度には十分に小さいことが言えます。

「最初、どの頂点間に辺が張られているか」や「S に点を追加したときに新しくどの辺が張られるか」などを愚直に求めようとすると、毎回 O(N+Q) だけの計算量がかかってしまい、容易に TLE します。ここで、あらかじめ平面を一片が d の正方形で区切っておくことを考えます。各頂点がどの正方形に含まれているのかを管理しておくことで、辺が張られる可能性がある頂点を高速に検索できます。具体的には、ある頂点 u との間に辺が張られる可能性がある頂点は、u を包含する正方形と、それと隣接する 8 個の正方形の、計 9 個の正方形に含まれる頂点のみです。「最終的に S に含まれる頂点同士のマンハッタン距離は $\frac{d}{2}$ 以上」という制約から、探索する頂点の最大値は高々 49 個であることが言え、これは十分高速です。このようなアルゴリズムをバケット法と言います。

今回の問題では、斜め 45 度回転をすることで、正方形の一片の大きさを d からさらに小さくすることができ、定数倍高速化を図れます。また、正方形全てを陽にもつと TLE or MLE してしまうため、map（Python なら dict）などで管理すると良いでしょう。

さて、上記より辺を張るパートが O(1) で処理できることがわかりました。あとは、出力クエリが飛んできたタイミングで、宝箱がある頂点のうち、いくつが原点と連結であるかを求められれば良いです。連結自体は UnionFind で管理でき、また、宝箱がある頂点のうちいくつと連結しているかも、頂点同士を merge するタイミングで一緒に管理ができます。

具体的には、以下のようにします。ans(i) を「i 番目の頂点が到達可能な宝箱の個数」とおきます。i が属する連結成分の代表元を r_i とおきます。頂点 i, j を merge したあとの i が属する連結成分の代表元を R_i とおきます。非連結な頂点 i, j を merge する際に、ans(R_i) を ans(r_i)+ans(r_j) で更新すればよいです。また、原点を 0 番目の頂点とおくと、出力クエリでは ans(r_0) を出力すればよいです。

以上で、この問題を O((N+Q)α(N+Q)) で解くことができました。ここで、α はアッカーマンの逆関数を表しています。

なお、時間計算量のオーダーはやや悪化しますが、バケット法のパートは kD-Tree で、その後のパートは並列二分探索や部分永続 UnionFind などでも解くことが可能です。