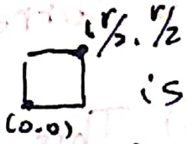
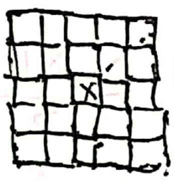
3. (a)

build the grid as the question suggested and use the lower left corner's coordinate to denote such a cell. e.g.  is $(0,0)$

Then for each point (x,y) we calculate $\lfloor x/r \rfloor$, $\lfloor y/r \rfloor$ and hash it using $h(\lfloor x/r \rfloor, \lfloor y/r \rfloor)$, if it collide with the hashed value, then we return true, else after hashing we check 24 of its surrounding cells  , if there is point that is in distance smaller then r, return true.

space-wise, there is at most n hash-value to store in the hash table, so it's size n or $O(n)$ generally

Complexity: For each point, hash is $O(1)$ and we know in all the surrounding 24 cells there is at most one point or otherwise we will have collsion in previous hashes, so it is still constant $O(1)$ checks so work per entry $O(1) \times O(n)$ entry $\Rightarrow O(n)$ in total

Correctness: the collsion in the hashing means ~~to~~ there are two points in the same cells and the longest distances is $\frac{\sqrt{2}}{2}r$ (diagonal) $\frac{\sqrt{2}}{2}r < r$ so we know ~~for~~ it's true $S(P)<r$, else we only need to check the surrounding 24 cells since it will take at least r to get cells outside these 24 cells.

Thus, it's constant amount of checks but also exhausted all the possibilities. So it's correct.

(b) This Algorithm is always correct since $S = \{C_{i,j}^? \mid 1 \le i < j \le 20\}$ it contains all pairs of P_i, P_j possible and the for loops check all the pairs in each iteration using closest-pair$(P_i \cup P_j)$, thus the Algorithm exhausted all the possible pairs and therefore will always be correct.

Complexity: since S is all possible pairs, the worst case is that each pair has a distance d_i and therefore there are $\frac{20 \times 19}{2} = 190$ distances. use C_i to denote we encountered the ith shortest distance, then we only need to call closest-pair once when it's C_1 ~~so $C_1 = 1$~~. for C_2, we need to call it twice. for C_3 it's $1 + \frac{1}{2}C_2 + \frac{1}{2}C_1$ since after 3^{rd} shortest we have 50% meet the shortest or the second shortest distance. generally it's

$$C_i = 1 + \frac{1}{i-1}\sum_{k<i} C_k$$ and our Expectation will be $\frac{1}{190}\sum_{i=1}^{190} C_i$

$$= \frac{1}{190}\sum_{i=1}^{190}\left(1 + \frac{1}{i-1}\sum_{k<i} C_k\right) = 1 + \frac{1}{190}\sum_{i=1}^{190}\frac{1}{i-1}\sum_{k<i} C_k$$

and if we expand $\frac{1}{i-1}C_k$ it's not hard to find out that

$$\frac{1}{i-1}\sum C_k = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{i-1} \le \ln(i-1) \text{ so.}$$

Scanned by CamScanner

Stirling approximation

$$1+\frac{1}{190}\sum_{i=1}^{190}\frac{1}{i-1}\sum_{k<i}C_k \le 1+\frac{1}{190}\sum_{i=1}^{190}\ln(i-1) = 1+\frac{1}{190}\ln 190! \approx 5.247<6$$

So the expected call of the subproblem is less than 6.

Then $T(n) \le 6T(n/10) + O(n)$ since we compared all the pairs

geometric series with $r<1$, converges!

So $T(n) \le O(n) + O(\frac{6}{10}n) + O((\frac{6}{10})^2 n) + \cdots = O(n)$

therefore the expected running time is $O(n)$