a)

Step 1: Min_Range (Town town) {
↳ Construct_Matrix [k][n]

Construct-Matrix([k][n], house_positions)
↳ for (i=0 to k) {
for (j=0 to n) {
if (i>=j) { range_Matrix.add[i][j] @f)}
↳ Base Case Base Stations > Houses

Lower Left Triangle is 0's. Base Case BS > Houses

else if (i==0 & j>0) {range Matrix[i][j] = $\frac{dj-d0}{2}$}
↳ 1 Base Station and n houses
Compute Range b/w House n & house 0. Insert into matrix

else if (i<j) {
minRange = {
take min of
Min { max(R_0, $\frac{d_n-d_1}{2}$)
max(R_1, $\frac{d_n-d_2}{2}$)
⋮
max(R_n, $\frac{d_n-d_n}{2}=0$) }

To find min_range for cell i,j, go to row i-1 and from Matrix[i-1][0---j] find max of current range and range of places 1 Base Station in remaining positions. Then take minimum of these

b)

Run Time: $\underbrace{O(k)}_{\text{Base Stations}} \cdot \underbrace{O(n)}_{\text{Houses}} \cdot \underbrace{O[n]}_{\text{min-Range finder}} = O(kn^2)$

c) OPTBSPosition(Town, town) {

for (i = 0 to k) {

for (j = 0 to n) {

Case 1 initialize lower left matrix: { if (i >= j) { Matrix[i][j] = house-Positions.get(0) if Matrix[i][j] = null

Case 2 initialize first row w/ $\frac{d(j)-d(0)}{2}$: else if (i == 0 && j > 0) { Matrix[i][j] = (dist(j) - dist(0) / 2)

else if (i < j) {

1) First find min Range. keep track of ~~Column~~ House where we found min Range. Also calculate max-Dist. Calculate Min { max(R_0, $\frac{d_n - d_1}{2}$) ~~max(R1~~ : max(R_n, $\frac{d_n - d_n}{2}$)

2) Then, using column of min range, ~~Calculate~~ find all max_distances in that row and column in matrix, and add to the new max distance, ~~to~~

3) Insert this ~~max~~ list of distances into matrix entry at (i, j)

d) $O(k) \cdot O(n) \cdot O(n) \cdot O(1) + O(k)$

Calculate Min Range — Calculate max Distance — Base station Location & Insert in matrix

$= O(k n^2 + O(k) = O(k)[n^2 + 1]$