# *Closest Pair Report*

*afin, alol, amand, anok, jave, jehj & thss*

*September 11, 2020*

## *Results*

Our implementation produces the expected results on all input–output file pairs. We see some floating point rounding errors in our results, however.
Our solutions runs through all the in 7.975 seconds.
See appendix for our results.

## *Implementation details*

First we sort all points on their x-axis. Then we recursively halve the array of points into smaller sub-arrays. We do this until there are 3 or less points in the sub-array of points, we return a floating point number $s$, which is the shortest distance between the points in that recursive call. The results are compared and the $s$ is returned.

After finding the smallest distance $s$, we use it to construct the 'split belt'. The belt will then have a width of $2s$. We find the points inside the split-belt by linearly searching from the split location, to the left and right in the points array.

We then sort the points in the split-belt by their y-axis, and call the same recursive function to find the smallest distance $s'$ between the points inside the belt. In order to save time we stop the search when there are 15 or fewer points inside the belt.

Finally we return the minimum of $s$ and $s'$

Our running time is $O(n \log n)$ for $n$ points.

## *Future Work*

In order to improve our solution we could we could use binary search instead of linear search when finding the points in the belt. This would be faster as binary search has a running time of $O(logN)$ instead of $O(N)$.

## *Alternative Implementation: kd-Tree Nearest Neighbour*

Another way of solving the closest pairs problem is to use a kd-tree with 2 dimensions. You can construct a kd-tree in $O(knlogn)$ time, where $k$ is the number of dimensions, and $n$ is the number of points. The running time for a nearest neighbour search is $O(logn)$ and iterating over all of the original points provides a solution in $O(nlogn)$ time just like our first proposed solution.

The difference in running time of the two implementations are solely on the linearithmic construction time of the kd-tree. To reveal the difference, we ran a shell-script that timed each algorithm over all input files several times.

While the first implementation is definitely a bit faster, the kd-tree implementation has the ability to find the specific distance one points and any other in $O(logn)$ time, which the first implementation only supports if the points are stored in sorted order by all axes, and a binary search is used to figure out where the comparison point is. That implementation would require $n^k$ memory, where $k$ is the number of dimensions. (There would be an array of points for each dimension, each containing points with $k$ values). If instead memory is to be saved, the recursive search for any two points would take $O(klogn)$ for $k$ dimensions over $n$ points.

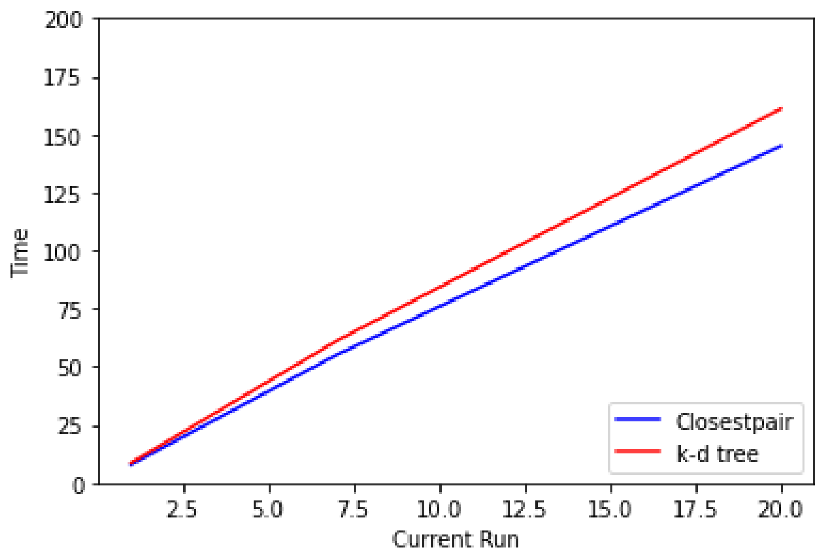

Figure 1: Runtime comparison

*Appendix*

```
data/a280-tsp.txt:        280     0.0
data/ali535-tsp.txt:      535     0.0
data/att48-tsp.txt:       48      131.46862743635836
data/att532-tsp.txt:      532     3.1622776601683795
data/berlin52-tsp.txt:    52      15.0
data/bier127-tsp.txt:     127     116.0
data/brd14051-tsp.txt:    14051   1.0
data/burma14-tsp.txt:     14      0.23000000000000043
data/ch130-tsp.txt:       130     0.6601809903932513
data/ch150-tsp.txt:       150     1.649216924729638
data/d1291-tsp.txt:       1291    24.999999999999773
data/d15112-tsp.txt:      15112   12.041594578792296
data/d1655-tsp.txt:       1655    25.399999999999636
data/d18512-tsp.txt:      18512   1.0
data/d198-tsp.txt:        198     22.936869882353186
data/d2103-tsp.txt:       2103    20.017242567346837
data/d493-tsp.txt:        493     17.96051224213805
data/d657-tsp.txt:        657     17.96051224213805
data/dsj1000-tsp.txt:     1000    679.9088174159826
data/eil101-tsp.txt:      101     1.4142135623730951
data/eil51-tsp.txt:       51      2.23606797749979
data/eil76-tsp.txt:       76      2.23606797749979
data/fl1400-tsp.txt:      1400    4.1719300090003735
data/fl1577-tsp.txt:      1577    8.35093407949055
data/fl3795-tsp.txt:      3795    4.1719300090003735
data/fl417-tsp.txt:       417     8.35093407949055
data/fnl4461-tsp.txt:     4461    10.0
data/gil262-tsp.txt:      262     1.0
data/gr137-tsp.txt:       137     0.7280109889280534
data/gr202-tsp.txt:       202     0.03999999999999915
data/gr229-tsp.txt:       229     0.3712142238654163
data/gr431-tsp.txt:       431     0.03999999999999915
data/gr666-tsp.txt:       666     0.022360679774997742
data/gr96-tsp.txt:        96      0.022360679774997742
data/kroA100-tsp.txt:     100     13.038404810405298
data/kroA150-tsp.txt:     150     13.038404810405298
data/kroA200-tsp.txt:     200     10.295630140987
```

| | | |
|---|---|---|
| data/kroB100-tsp.txt: | 100 | 26.0 |
| data/kroB150-tsp.txt: | 150 | 8.06225774829855 |
| data/kroB200-tsp.txt: | 200 | 5.0 |
| data/kroC100-tsp.txt: | 100 | 17.72004514666935 |
| data/kroD100-tsp.txt: | 100 | 11.661903789690601 |
| data/kroE100-tsp.txt: | 100 | 22.135943621178654 |
| data/lin105-tsp.txt: | 105 | 31.0 |
| data/lin318-tsp.txt: | 318 | 31.0 |
| data/linhp318-tsp.txt: | 319 | 31.0 |
| data/nrw1379-tsp.txt: | 1379 | 2.8284271247461903 |
| data/p654-tsp.txt: | 654 | 15.0 |
| data/pcb1173-tsp.txt: | 1173 | 1.0 |
| data/pcb3038-tsp.txt: | 3038 | 1.0 |
| data/pcb442-tsp.txt: | 442 | 50.0 |
| data/pla33810-tsp.txt: | 33810 | 930.3897032964197 |
| data/pla7397-tsp.txt: | 7397 | 930.3897032964197 |
| data/pla85900-tsp.txt: | 85900 | 728.0109889280518 |
| data/pr1002-tsp.txt: | 1002 | 100.0 |
| data/pr107-tsp.txt: | 107 | 200.0 |
| data/pr124-tsp.txt: | 124 | 150.0 |
| data/pr136-tsp.txt: | 136 | 170.0 |
| data/pr144-tsp.txt: | 144 | 100.0 |
| data/pr152-tsp.txt: | 152 | 75.23961722390672 |
| data/pr226-tsp.txt: | 226 | 100.0 |
| data/pr2392-tsp.txt: | 2392 | 1.0 |
| data/pr264-tsp.txt: | 264 | 100.0 |
| data/pr299-tsp.txt: | 299 | 1.0 |
| data/pr439-tsp.txt: | 439 | 90.13878188659973 |
| data/pr76-tsp.txt: | 76 | 300.0 |
| data/rat195-tsp.txt: | 195 | 6.324555320336759 |
| data/rat575-tsp.txt: | 575 | 2.0 |
| data/rat783-tsp.txt: | 783 | 1.0 |
| data/rat99-tsp.txt: | 99 | 4.123105625617661 |
| data/rd100-tsp.txt: | 100 | 4.796664361824801 |
| data/rd400-tsp.txt: | 400 | 1.1739771249901285 |
| data/rl11849-tsp.txt: | 11849 | 9.0 |
| data/rl1304-tsp.txt: | 1304 | 32.0 |
| data/rl1323-tsp.txt: | 1323 | 48.0 |
| data/rl1889-tsp.txt: | 1889 | 32.0 |
| data/rl5915-tsp.txt: | 5915 | 32.0 |
| data/rl5934-tsp.txt: | 5934 | 9.0 |

| | | |
|---|---|---|
| data/st70-tsp.txt: | 70 | 1.0 |
| data/sw24978-tsp.txt: | 24978 | 0.2776999999987311 |
| data/ts225-tsp.txt: | 225 | 500.0 |
| data/tsp225-tsp.txt: | 225 | 6.5 |
| data/u1060-tsp.txt: | 1060 | 70.68239950652483 |
| data/u1432-tsp.txt: | 1432 | 100.0 |
| data/u159-tsp.txt: | 159 | 100.0 |
| data/u1817-tsp.txt: | 1817 | 25.389999999999873 |
| data/u2152-tsp.txt: | 2152 | 25.389999999999873 |
| data/u2319-tsp.txt: | 2319 | 100.0 |
| data/u574-tsp.txt: | 574 | 2.941598205057742 |
| data/u724-tsp.txt: | 724 | 3.3199999999999363 |
| data/ulysses16-tsp.txt: | 16 | 0.3883297567789516 |
| data/ulysses22-tsp.txt: | 22 | 0.08944271909999127 |
| data/usa13509-tsp.txt: | 13509 | 2.7770000000018626 |
| data/vm1084-tsp.txt: | 1084 | 22.0 |
| data/vm1748-tsp.txt: | 1748 | 22.0 |