# COMP3323: Advanced Database Systems

## Assignment 3 Report (6 Pages)

***Q. Please write down your experiment environments (e.g., machine used, CPU speed, disk speed, main memory sizes).***

A.

- **Machine used:** MacBook Pro 13-inch (2018)
- **CPU speed:** 2.3 GHz
- **Disk speed:**
  - Write speed: 2,627MB/s
  - Read speed: 2,505MB/s
- **Main memory sizes:** 8 GB 2133 MHz

The code was written in Java 8 and the IDE used was Eclipse.

*Q.* ***Why is it possible to skip the cell c if dlow(c)>t?***
*(where dlow(c) smallest distance from q to cell c, t is the k-th largest distance in the k-NN set)*

A. We are trying to find the nearest k neighbours for point q. As we know that t is currently the distance of neighbour that is furthest away from q, if the shortest distance between cell c and point q is greater than t, then any point inside the cell c has to have distance greater than t and hence, the whole cell c can be ignored.

*Q.* ***Explain why pruning an entire layer of cells is sufficient to terminate the algorithm?***

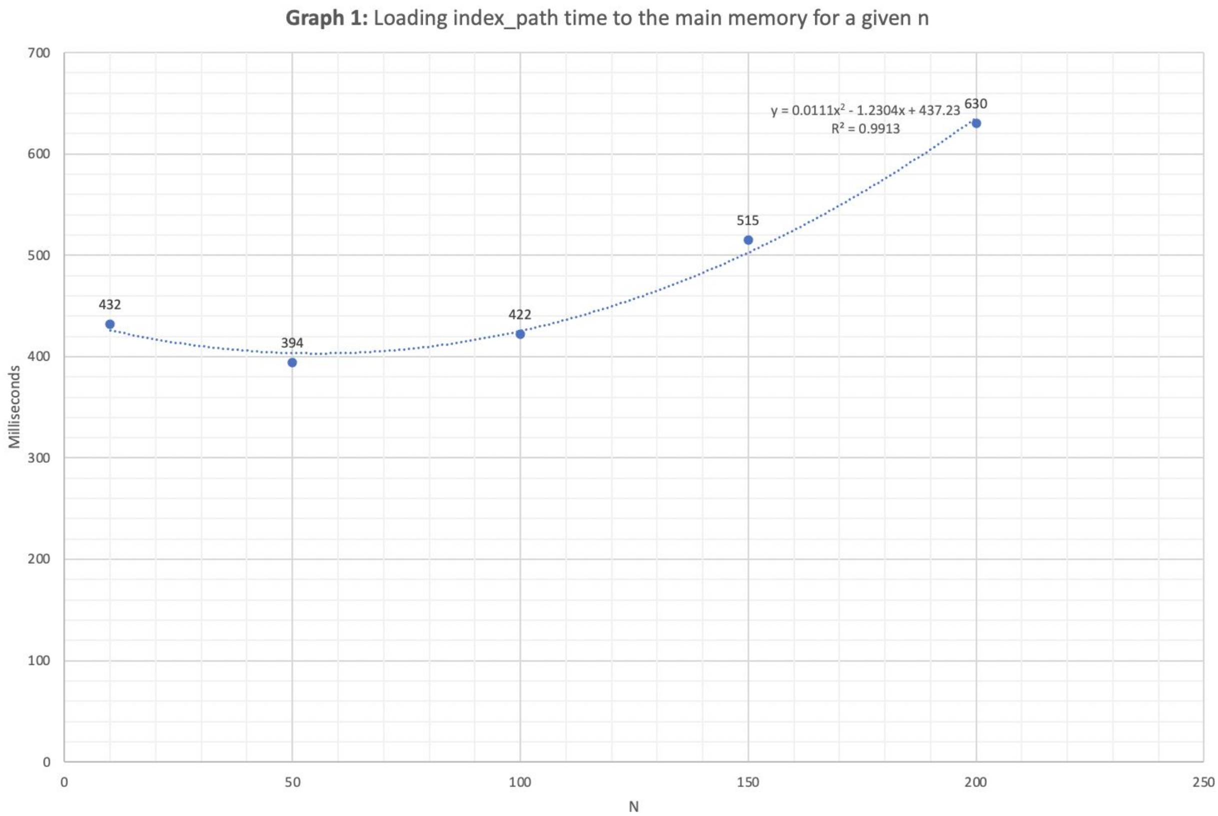
A. If we have checked a whole layer with no cell satisfying the condition of dlow(c) <= t, then, any next layer would only be further away from point q. Hence, we can terminate the whole algorithm.

*Q.* ***Perform the following experiments: (Hint: you can combine part c and d into one graph.)***

A. Generate 100 random query points within maxBox.

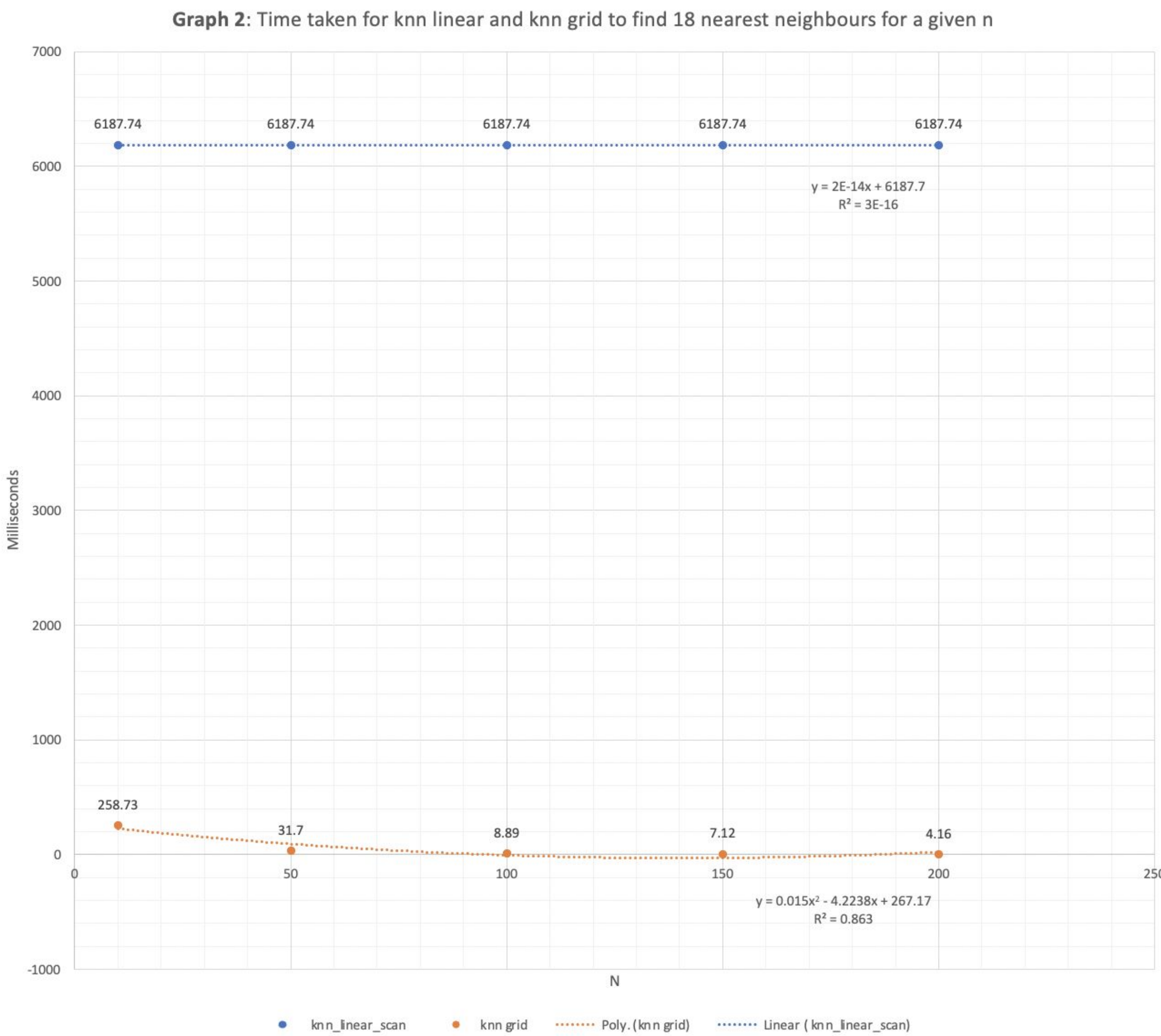
   *Please see the* ***appendix***

B. <u>Plot a line graph to show the time of loading “index_path” to the main memory, the following value of n: 10, 50, 100, 150, 200.</u>



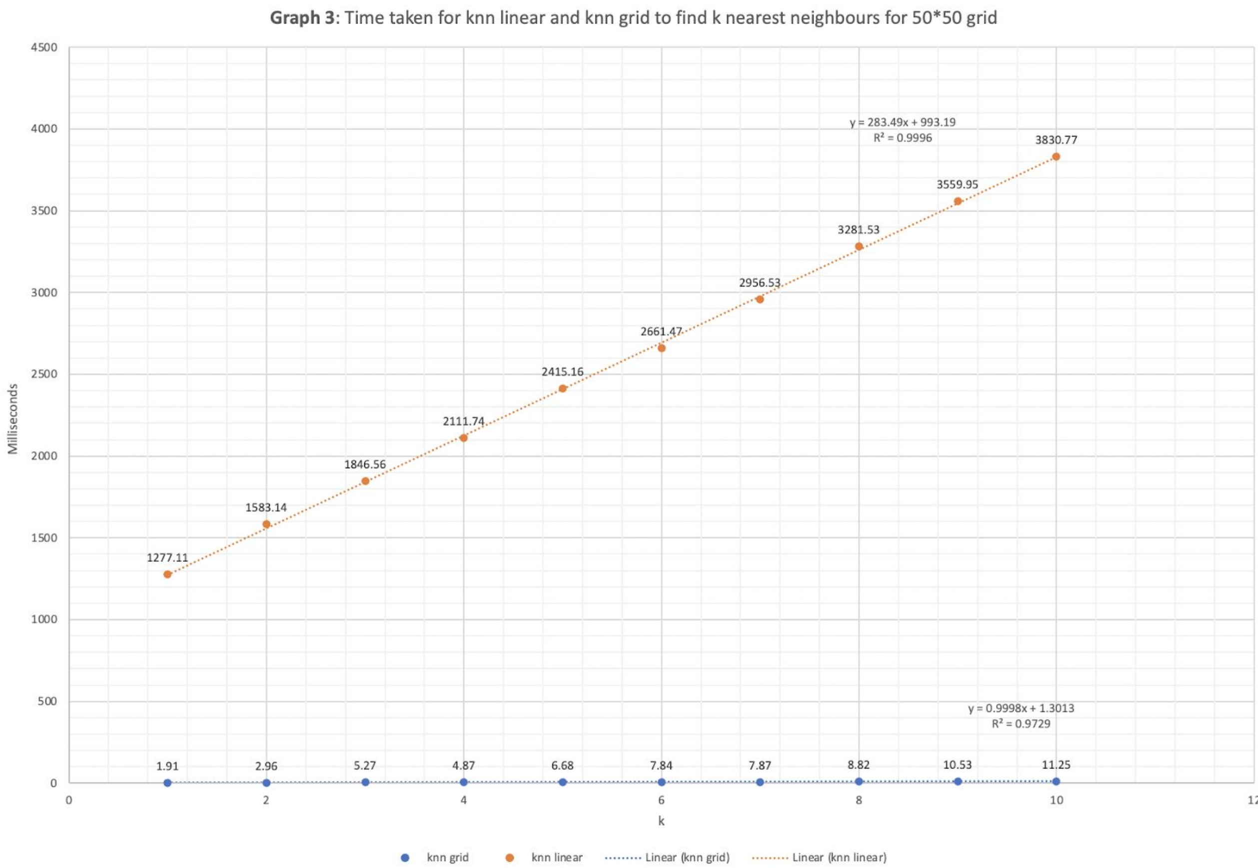

A. The structure used for “index_path” is a 2D arraylist ( e.g. Cell [Row][Column] - see Fig 1 ). For small values of n (e.g. 10), we have relatively few references to the 1D array containing many points. Hence, it takes more time to load “index_path”. But as you keep increasing the value of n, the references are increasing slightly compared to the reduction in the size of 1D arraylist containing the points and hence, the time to load decreases. After n = 50, the trade off of increasing references to decrease 1D arraylist sizes is lost and increasing references not only adds more time but does not decrease the size of the 1D array significantly. This results in the value of n greater than 50 to take more time to load “index_path”.

C. <u>Plot two line graphs by reporting the average execution times of knn_grid and knn_linear_scan for the query points generated in (a), against the following value of n: 10, 50, 100, 150, 200. For knn_grid, DO NOT consider the time for loading “index_path” to the main memory.</u>




A. For a given value of k (k=18), knn linear scan would stay constant as it goes through the whole data regardless of the value of n to find the nearest neighbours. However, as n is increased for knn grid, the size of each cell as well as the average number of points inside each cell also decreases. Thus, knn grid has to go through less number of points which satisfies the pruning condition and hence, it takes less time as n increases. Please keep in mind that at a certain big number of n, the cell would start becoming too small and hence, eliminate the advantage talked above, leading to increased time as n increases. For the given graph, the knn grid is at least 30x faster than the knn linear.

D. <u>Plot two line graphs by reporting the average execution times of knn_grid and knn_linear_scan for the query points generated in (a), against the following value of k=1, 2,.., 10. For knn_grid, DO NOT consider the time for loading "index_path" to the main memory.</u>




A. For the given graph, knn grid is at least 600x faster than knn linear. For the fixed value of n to be 50, Knn linear increases linearly as value of k increases. This is because time taken to go through the whole data stays the same however, as more neighbours need to be kept and checked for each new point. Similarly, the knn grid also increases linearly as the value of k increases. This is because higher value of k leads to checking more cells to satisfy this high number of k, leading to increased time taken.

## *Appendix*

*X-value Y-value*

*79.63165936958941 127.98645384595159*
*39.33161639707677 -81.43826531948315*
*-84.89995431034819 -148.265881514743*
*22.073668288721365 -93.35268886818739*
*6.183437088640204 -86.2277421600778*
*-81.05785727958586 -117.17496007772093*
*-15.052808421291076 106.1729238060343*
*-61.90498288486227 105.52782588082852*
*-88.8736416155343 123.84010614990513*
*-3.3567615959766215 -18.342074347120672*
*-69.52502344108981 121.25662852812059*
*-1.3988101471996117 -108.02613303661244*
*84.20571634189292 22.814605452877657*
*-66.65932777797018 166.17597123832041*
*86.69087007963944 170.44778884575646*
*65.67442960835453 175.7860449231746*
*-72.71289560651202 30.052811594256553*
*21.449301405902986 -169.46095471924644*
*23.962887469491648 155.9118542698128*
*28.40782924253989 139.48271325800374*
*-58.651310904465035 50.01044422071206*
*-48.91878356044242 -100.88419470176835*
*83.89676397716153 128.9818556797008*
*-39.394694756836 135.08642104315112*
*46.0284282963635 -66.79631664525768*
*-74.39477586971266 43.21883019786969*
*45.46336806018485 -43.277917000729644*
*78.67325008133616 98.14078206480934*
*23.366841351134 -164.21410344484153*
*-15.53170678836311 139.14252315975347*
*-74.48302299582764 -161.36465305071513*
*70.09442058711963 158.31492773617975*
*2.385960165415227 103.20906311607484*
*-83.83041527733822 -45.71416048554815*
*42.8450627656087 -71.99161625714702*
*-72.12521851614844 -45.797551214818924*
*26.145101249275427 38.089257128632426*
*29.277698125204736 -18.408289006083805*
*39.89018563987128 -122.56016889256564*
*-28.646193367997228 -29.4280781861263*
*-52.085959374436406 -45.954231299044864*
*34.381770155787365 62.585450271963424*
*21.28463962422751 -75.85461014387697*
*-61.23100398614156 167.37608651551983*
*48.10425993729763 37.93343867781559*
*-5.133005658165985 -42.45608508641314*
*-6.6615488992208896 -87.9153241637263*
*-38.034879602377124 103.27657194822143*
*-71.40111014571605 -108.46232719315331*
*-30.594722696007523 95.17763293281382*

*-46.37859712769347 166.20428948966463*
*-21.361108637473606 -108.30859627997259*
*-53.8780546368625 -118.1482972152694*
*54.05158233991017 115.05783514732457*
*-35.220356398257806 85.14167375264822*
*-57.49034688687805 0.6220982239136106*
*-48.71251961806841 -143.2943950575374*
*56.83661337931807 -19.52951681911125*
*-48.27244397066936 -40.21583109020986*
*-3.1932826786075594 118.767578413541*
*44.96931931555821 69.7666413671057*
*62.7464635665294 36.62924620988156*
*86.35550168569753 -61.872004073612985*
*1.4639682702598407 -124.51822211344911*
*8.631967512042308 48.58042631108853*
*-48.02015196740752 15.629565389226542*
*-31.105960968070953 51.73153206110203*
*9.412398429147785 44.982122954040705*
*50.263311297233315 72.77245896232228*
*-60.626442816854976 -91.62427521417183*
*80.0752957767954 0.5074336710762282*
*-87.63886397893174 123.26575155493231*
*-27.45021937334384 111.57952279017456*
*-72.11325395910343 51.363033486640944*
*23.232034634726105 -42.35175847965803*
*-58.54236590038462 -69.95937030229231*
*34.93453004666003 3.3305531689304075*
*14.034683898844875 -12.55284894190001*
*0.47514289553745925 -146.87012674898227*
*-26.78235963716972 -114.17142018686772*
*-13.800660833157707 148.54643922557761*
*28.162586528675845 -6.217483956787021*
*-47.9099243182115 -10.31237485196624*
*-43.88443776296394 9.847178727254885*
*-10.789218369637226 -46.021667382622695*
*-2.5498300473973075 59.762007205607915*
*13.292933643899104 -80.99086205883934*
*38.22585852754702 -104.4723786452181*
*4.4713704896485496 -155.44606817792746*
*44.53754668198886 31.239784026760162*
*47.48678141186315 24.138925050951684*
*-78.69391576070859 -113.8243999173458*
*4.1162115397459615 11.851370002519076*
*79.66305917456211 126.36942298561621*
*20.158859390496048 82.3389944668179*
*71.71220779431198 -130.067854308954*
*12.99588190455141 123.97016712054489*
*49.20375053873008 55.43924077051341*
*-37.440767814862305 138.89999132284453*
*83.53681517924232 -14.758464862062453*