Informe

Grau A– Q1 Curs 2023-2024
Departament de Ciències de la Computació
Universitat Politècnica de Catalunya
Autors: Alex Bueno, Álvaro Terrón, Oriol Ramos i Oriol Roca

1 Primera part: Informe tècnic

1.1 Disseny i implementació dels arbres k-dimensionals

Classe KDTree

- *struct Punt*: Struct que representa el conjunt d'atributs que pertanyen a la clau de cada node de l'arbre *KDTree.*
  - *vector<double> coords*: Vector de reals $\in$ [0,1] de mida k que representa els atributs de la clau.
  - *Punt(vector<double> coords)*: Constructora del struct *Punt* amb el conjunt d'atributs *coord.*
  - *double distanciaEucl(const Punt& p2) const*: Funció que retorna la distància Euclidiana entre el *Punt* implícit (*this*) i el *Punt p2.*

- *struct Node*: Struct que representa un node de l'arbre binari *KDTree.*
  - *Punt punt*: Clau del *Node* representada per els seus atributs.
  - *Node* esq*: Punter al subarbre de l'esquerra.
  - *Node* dre*: Punter al subarbre de la dreta.
  - *Node(Punt punt)*: Constructora del struct *Node* amb el *Punt punt* com a clau.

- *KDTree():* Constructora de l'arbre *KDTree* buit.

- *void afegirNode(const Punt& p):* Afegeix en el lloc corresponent el *Node* amb clau *p* al arbre en base als valors del *Punt p.*

- *void borrarArbre(node* r):* Elimina tots *Nodes* de l'arbre recursivament alliberant la memòria.

- *void escriuArbre():* Mostra per pantalla l'arbre en preordre.

- *int getNodesVisitats():* Retorna el número de *Nodes* que visita la funció *veiMesProper().*

- *Node* getRoot():* Retorna el *Node* arrel de l'arbre.

- *Punt veiMesProper(const Punt& objectiu):* Retorna el *Punt* més proper al *Punt objectiu.*

- *void generarArbreAleatori(int k, int n):* afegeix n nodes aleatoris al arbre de dimensió k

- **Atributs:**
  - *int nodes_visitats*: Nodes visitats per la funció *veiMesProper()* que s'actualitza a mesura que s'executa.
  - *double millorDistancia*: Distància més propera al *Punt* objectiu que es va actualitzant a mesura que s'executa la funció *veiMesProper().*
  - *Node* millorPunt*: Node més proper al *Punt* objectiu que al mateix temps que millorDistancia es va actualitzant durant l'execució de *veiMesProper().*
  - *Node* root*: Node arrel de l'arbre.

1.2 Algoritme del veí més proper

La funció *veiMesProper(objectiu)* fa una crida a la funció auxiliar *searchNearest* inicialitzant *millorPunt* a null. La funció s'encarregarà de donar-li valor com explicarem a continuació.
void searchNearest (Node node, const Punt& objectiu, int prof)*

La funció *searchNearest(node, objectiu, prof)* calcula el veí més proper del node objectiu començant des de l'arrel i de manera recursiva.

1. **Comencem** amb el node arrel de l'arbre.

2. **Cerca Descendent**:
   - Si l'arbre està buit, retornem null indicant que no hi ha veí.
   - Calculem la distància entre el punt actual (node de l'arbre) i el punt de cerca.
   - Actualitzem *millorPunt* i *millor_Distancia* en cas que la distància inicial sigui millor que les guardades en aquell moment.

3. **Decidim el Fill d'Exploració**:
   - Calculem el discriminant segons la profunditat actual en la que ens trobem i decidim si explorar fill dret o fill esquerre segons la distància del node actual a l'objectiu fent servir només aquest discriminant. (*discr = prof%k*).

4. **Explorem el Fill**:
    - Explorem el fill que correspongui al punt de cerca en base al discriminant actual *searchNearest(subarbreProp, objectiu, prof+1).*
    - Al tornar de l'exploració, comparem la distància entre *millorPunt* i el punt amb la distància entre el punt i la línia de l'espai representada pel discriminant actual. (Explicació al punt 6).

5. **Recursió**:
    - Continuem el procés de cerca recursiva fins arribar a una fulla de l'arbre (un node que no té fills).

6. **Retrocedim**:
    - Al retrocedir des dels nodes fins a l'arrel, verifiquem si pot haver-hi algun node al subarbre oposat de la ubicació que podria ser més proper. Per fer-ho, comparem la distància entre *millorPunt* i el punt amb la distància entre el punt i la línia de l'espai representada pel discriminant actual *searchNearest(subarbreLlun, objectiu, prof+1).*

7. **Nodes visitats**:
    - Cada vegada que entrem a la funció incrementem el valor de *nodes_visitats*, de manera que tenim el comptatge de tots els nodes que visitem durant la cerca del veí més proper.

1.3 Experimentació

Sabem que el cost de l'algorisme *veiMesProper* és $\Theta(n^{\zeta} + \log n)$. Per tant la complexitat temporal/espacial de l'algorisme ve donada per el coeficient ζ i el nombre de nodes ***n***. L'objectiu de l'experimentació era estimar experimentalment el valor de ζ a partir de múltiples execucions de l'algorisme per diferents n's, k's i diferents arbres aleatoris. I a partir del nombre de nodes de l'arbre i els visitats per l'algorisme, trobar el valor de ζ en cada execució i estimar-lo a partir d'una mitjana.

<u>Metodologia de l'experimentació</u>

- ***n:*** Nombre de nodes de l'arbre. Per obtenir dades representatives, hem decidit realitzar proves per **41 valors diferents de n**; de 1,000 a 100,000 nodes, incrementant cada n diferent en 2500 nodes. D'aquesta forma tenim el nombre de nodes visitats per múltiples valors d'n des d'un nombre relativament petit de nodes, com és 1,000 fins a un nombre suficientment gran com és 100,000.

- ***k:*** Dimensió de cada Punt del KDTree. Per realitzar l'experimentació s'ha decidit executar l'algorisme per valors de la dimensió **k de 2 a 6**, com s'especificava a l'enunciat de la pràctica, experimentant així amb diferents valors de k de tamany raonable per les seves possibles aplicacions.

- ***N:*** Nombre d'arbres aleatoris diferents. En aquest cas, hem decidit establir la **N = 50** per generar suficients arbres com per cobrir diferents nivells de desequilibri i que la mostra sigui més exhaustiva sense arribar a trigar massa l'execució de l'experiment. L'objectiu era obtenir valors representatius amb un nombre d'arbres ni massa gran ni massa petit, i hem arribat a la conclusió que 50 és un valor suficientment exhausitu a nivell d'observar diferents possibilitats.

- ***Q:*** Nombre de consultes per cada arbre. Fixem el valor **Q = 50**, llavors per cada arbre es faran 50 consultes de veïns més propers amb punts aleatoris; el valor pretén cobrir la generació de diferents punts els quals tinguin diferents profunditats a l'arbre que ens puguin donar valors extrems de la mostra, i igual que en la decisió de la constant N, no hem volgut passar-nos ni de llarg ni de curt i hem decidit que 50 és un valor representatiu, ja que pot representar molts nivells de profunditat diferents.

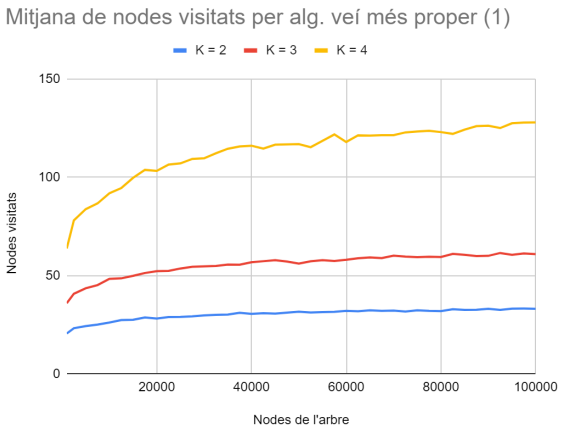
Un cop hem definit les diferents variables, el procés d'experimentació ha consistit en què, per cadascun dels 41 valors diferents de n, dins de cada valor de k de 2 a 6, el programa realitza 2,500 execucions de l'algorisme (50 consultes per 50 arbres aleatoris diferents). Per tant, per cada execució obtenim la mitjana del número de nodes visitats per l'algorisme.
Una vegada obtingudes les dades de les mitjanes, es poden calcular per cada combinació de {n,k} els valors de ζ (tenim que el cost $\Theta(n^{\zeta} + \log n) = Xi$, on Xi és cada mitjana de nodes visitats per cada conjunt de valors de les 50 querys dels 50 arbres, i per tant es pot aïllar ζ i trobar cada valor per posteriorment, calcular la mitjana.

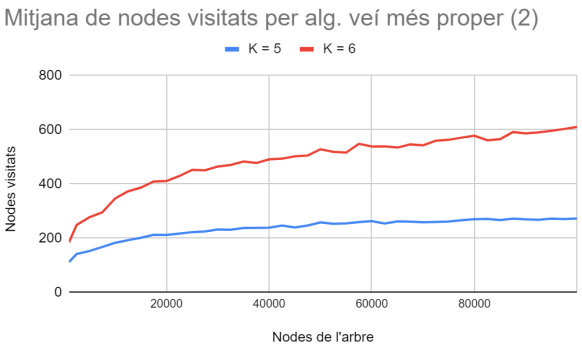
RESULTATS

EXPERIMENT 1: MITJANA I VARIÀNCIA DELS NODES VISITATS AMB AUGMENT DE NODES DE L'ARBRE.

El primer experiment realitzat és comprobar a partir de la formula $\Theta(n^{\zeta} + \log n)$ suposant que n^{ζ} és constant per qualsevol dimensió, que el nombre de nodes visitats per l'algorisme de búsqueda té un creixement logarítmic i que el cost del algoritme de búsqueda de veí més proper no es lineal.

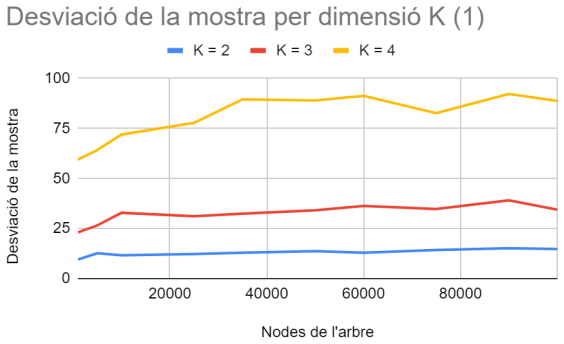
Hem separat els resultats en dues gràfiques diferents ja que si ajuntem les funcions des de k = 2 fins a k = 6, el creixement de les k més grans (5 i 6 especialment), pràcticament no deixen veure el creixement de les k més petites, per tant hem decidit separar les més petites (Mitjana de nodes visitats per l'algorisme del veí més proper (1)) en una i les més grans en una altra (Mitjana de nodes visitats per l'algorisme del veí més proper (2)) per una millor visualització de les dades.

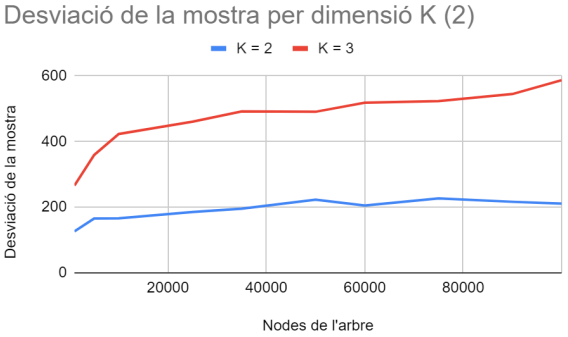




En els gràfics podem observar que per cada dimensió el valor de nodes visitats amb arbres d'una mateixa mida creix, per el que podem suposar que el valor C augmenta amb la dimensió. També s'observa que el creixement de nodes visitats no es lineal amb el augment de la mida així que podem afirmar que el algorisme és bastant eficient (per arbres de 100.000 nodes ens troba en mitjana el veí més proper explorant només uns 600 nodes).

En l'apèndix de la documentació del projecte es mostren uns gràfics on es veu la variància de la mostra (2), per comprovar que la generació d'arbres ha creat aquestes situacions extremes que complementen els resultats obtinguts i al donar uns valors tan grans ens confirmen que els valors escollits de N i Q son suficients com per generar una mostra representativa.





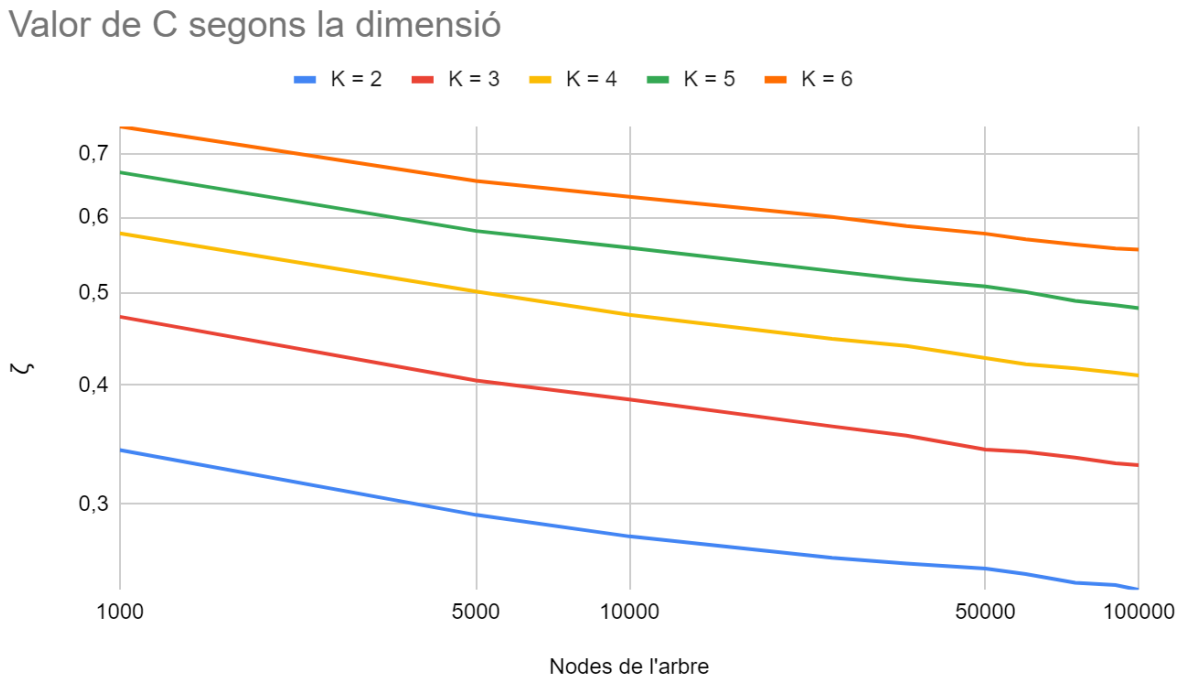
EXPERIMENT 2: ESTIMACIÓ DE ζ

Per l'estimació del valor de la constant hem agafat 10 valors de n i hem aplicat la següent fórmula per substituir i obtenir una estimació aïllant el valor de ζ desde la fórmula inicial.

$$\zeta = \frac{log_{10}(MITJANA - log_2 MIDA\ ARBRE)}{log_{10}(MIDA\ ARBRE)}$$

La idea és aplicar aquesta fórmula amb els valors de la mitjana obtinguts per els valors de mida de l'arbre i al substituirlos a la fórmula ens hauria de donar un valor entre 0 i 1, i que segons la dimensió sigui més gran.

PLOT LOG C_n / LOG N



Veiem com per totes les dimensions segueix una funció logarítmica inversa, i hem observat que segons aumenta el número de nodes de l'arbre, disminueix la constant de manera lleu, i ademés s'estabilitza (disminueix considerablement la variància). És per això que hem vist oportú agafar com a referència el valor més petit de la constant, que com hem explicat és justament on tenim el número de nodes més gran, ja que serà la millor aproximació del valor que volem obtenir. Per tant els nostres valors finals de ζ son els que ens dona els arbres de mida 100.000:

Dimensió	2	3	4	5	6
ζ	0,243696	0,329451	0,409385	0,481584	0,554776

1.4 Conclusions

Com a conclusió d'aquest experiment podem afirmar que efectivament es compleix que el cost de l'algorisme per trobar el veí més proper d'un node en un arbre *k*-dimensional és $\Theta(n^{\zeta} + \log n)$ ja que es pot observar en les gràfiques mostrades anteriorment que el nombre de nodes visitats a l'executar l'algorisme per a tot *k* entre 2 i 6 s'aproxima bastant al comportament teòric de la funció logarítmica, tot i que no arriba mai a ser exacte atès que no podem assegurar que el nostre arbre *k*-dimensional sigui sempre equilibrat, degut a la forma en que és creat.

També podem observar clarament gràcies a les gràfiques que el valor de ζ depèn en gran mesura del valor de la dimensió *k*, ja que veiem que a l'augmentar el valor de *k* també incrementa el valor de ζ, per a qualsevol mida en nodes de l'arbre, mentres que si reduïm el valor de *k* podem observar que el seu valor es decrementa, tot i que mai arribarà a ser igual a 1 ni més petit que 0.

A més a més, també podem arribar a veure que per a tot valor de *k* amb els que hem dut a terme l'experiment, el valor de ζ és major per a arbres força petits i que conforme es va augmentant el tamany de l'arbre, el seu valor es va decrementant i cada vegada sembla que vagi guanyant més estabilitat i menys variació.

2 Segona part: Descripció i valoració del procés d'autoaprenentatge

2.1 Metodologia

En primer lloc, un cop vam entendre tots l'enunciat i ja sabíem exactament el que havíem de fer, ens vam dividir la feina en dos subgrups: un s'encarregaria de pensar una forma d'implementar el *k-d tree* i l'altre, de començar a pensar l'algorisme per trobar el veí més proper i posteriorment posar en comú les idees que havíem tingut per tal d'arribar a un acord entre tots els integrants. D'aquí endavant, un cop ja sabíem com plantejar més o menys tots dos aspectes, ens vam centrar en la implementació del *k-d tree.*

Per fer-ho, vam estar investigant principalment les implementacions que havíem vist anteriorment dels arbres binaris (*BinTree*) en assignatures passades, com ara PRO2, i vam adaptar-les per tal d'aconseguir la nostra versió d'aquests arbres amb nodes k-dimensionals; i un cop ja la vam tenir tota acabada ens vam centrar complemetament en l'algorisme de cerca del veí més proper, que amb la idea en ment no va suposar gaire difícil de programar.

Per la següent part del projecte, l'experimentació, vam estar discutint tots quatre els valors que cadascun de nosaltres consideràvem més adients per dur a terme l'experiment tot argumentant el nostre punt de vista i avaluant de forma experimental cadascuna de les opcions proposades fins que finalment ens vam posar tots d'acord, tots vam començar a recollir les dades que necessitàvem i a continuació vam fer servir l'*R*, una eina molt usada en estadística, per trobar els valors del cost mitjà i la variància que es busquen.

Finalment, una vegada ja havíem obtingut tots els valors del cost mitjà i variància, ens vam disposar a calcular els diferents valors de ζ i a crear les gràfiques per extreure les nostres conclusions del treball.

En definitiva, durant tot el treball hem sabut organitzar-nos, distribuir-nos la feina de forma equitativa i ajudar-nos els uns als altres, així com prendre totes les decisions en equip.

2.2 Valoració del procés d'autoaprenentatge

En quant a la part d'autoaprenentatge inicialment ens vam haver de familiaritzar amb els arbres *k-d trees*, buscant informació de la seva utilitat i de les seves possibles implementacions.

Gràcies a què vam decidir basar-nos en implementacions d'arbres binaris de cerca vistes a PRO2 per implementar la nostra classe, hem après a reutilitzar codi per tal d'aplicar-ho a noves funcionalitats i pensar en els canvis necessaris per poder resoldre el nostre problema.

Respecte a l'algorisme, valorem haver après la utilitat d'aquests tipus d'algorismes, ja que es poden veure totes les aplicacions pràctiques que tenen i com apareixen en recursos que utilitzem diàriament així com el procés de pensar en com implementar l'algorisme i la cerca d'informació per tal de fer-ho. Ademés ens ha proporcionat una comprensió detallada dels conceptes de cerca i comparació de punts en un espai multidimensional, la qual cosa, com es deia a l'exemple, té aplicacions pràctiques molt útils com el Google Maps.

D'altra banda, el procés d'experimentació amb el nostre propi algorisme ha sigut una bona forma pràctica d'aprendre a obtenir informació real i detallada a partir de múltiples mostres preses del nostre programa, aconseguint així experiència en recollida de dades d'experiments propis i reals. Hem hagut d'experimentar en com executar quan havíem trobat la Q i la N perquè no trigués massa.

3 Tercera part: Bibliografia

What is the k-nearest neighbors algorithm? (s/f). Ibm.com. Recuperat el 30 de setembre de 2023

https://www.ibm.com/topics/knn

K vecinos más próximos. (s/f). Wikiwand. Recuperat el 1 d'octubre de 2023, de https://www.wikiwand.com/es/K_vecinos_m%C3%A1s_pr%C3%B3ximos

Copy-based kd-trees y quad-trees. (s/f). Upc.edu. Recuperat el 10 d'octubre de 2023 https://upcommons.upc.edu/bitstream/handle/2099.1/9078/PFC_Copybased.pdf?sequence=1&isAllowed=y

Varianza y desviación típica en R. (s/f). R CODER. Recuperat el 15 d'octubre de 2023

https://r-coder.com/varianza-desviacion-tipica-r/?utm_content=cmp-true

BinTree.hh. (2021). Professorat de PRO2. Recuperat el 2 d'octubre de 2023 https://github.com/AlexBuenoL/BinTree-PRO2/blob/main/BinTree.hh

KD-Tree Nearest Neighbor Data Structure. (2020). *Stable Sort* on Youtube. Recuperat el 8 d'octubre de 2023

https://www.youtube.com/watch?v=Glp7THUpGow

4 Quarta part: Apèndixs

(1) Taula de les mitjanes del número nodes visitats en les búsquedes del veí més proper

NodesVisitats \ k	2	3	4	5	6
1000	20,5632	35,9724	63,8628	111,48	184,566
2500	23,2432	40,8156	78,1372	141,431	248,79
5000	24,329	43,598	83,865	152,636	277,25
7500	25,1108	45,2128	86,7944	167,398	295,204
10000	26,152	48,3648	91,9812	182,771	346,57
12500	27,4224	48,6372	94,6116	192,431	372,756
15000	27,5376	49,91	99,7924	201,341	386,484
17500	28,7384	51,3544	103,855	212,31	409,093
20000	28,2016	52,3236	103,342	211,773	410,955
22500	28,944	52,4432	106,542	216,787	429,14
25000	28,9992	53,6532	107,186	222,25	452,091
27500	29,28	54,5652	109,44	224,519	450,37
30000	29,7788	54,7224	109,746	232,26	464,46
32500	30,0572	54,9844	112,343	231,262	470,251
35000	30,2364	55,6652	114,616	237,309	482,814
37500	31,1028	55,5664	115,761	238,208	477,365
40000	30,5868	56,8628	116,165	238,386	490,668
42500	30,8796	57,3704	114,678	246,225	493,301
45000	30,6824	57,9212	116,669	239,4	502,09
47500	31,1764	57,19	116,809	246,415	504,939
50000	31,6588	56,1404	116,974	258,152	528,018
52500	31,234	57,35	115,412	252,684	518,52
55000	31,4952	57,8908	118,634	254,298	515,73
57500	31,586	57,5092	121,896	259,131	548,165
60000	32,0944	58,0964	118,06	262,911	538,137
62500	31,898	58,8648	121,417	253,784	538,648
65000	32,3724	59,314	121,312	261,894	534,305
67500	32,0736	58,9412	121,532	260,648	545,946
70000	32,2484	60,2008	121,542	258,976	542,615
72500	31,7392	59,6996	122,946	259,81	559,56
75000	32,356	59,4056	123,384	261,631	563,014

NodesVisitats \ k	2	3	4	5	6
1000	20,5632	35,9724	63,8628	111,48	184,566
2500	23,2432	40,8156	78,1372	141,431	248,79
77500	32,102	59,6608	123,731	266,37	570,928
80000	31,9828	59,4948	123,09	270,122	577,927
82500	32,9268	61,1112	122,198	270,892	561,142
85000	32,5604	60,6408	124,338	266,772	565,288
87500	32,6336	59,98	126,124	272,267	591,694
90000	33,1128	60,1088	126,319	269,316	586,725
92500	32,592	61,5524	125,154	267,61	590,589
95000	33,238	60,642	127,574	271,904	596,154
97500	33,2584	61,3384	127,921	270,692	602,89
100000	33,1476	60,9964	128,02	272,42	610,736

(2) Taula de les variàncies del número nodes visitats en les búsquedes del veí més proper

NodesVisitats \ k	2	3	4	5	6
1000	90,79464	533,0768	3531,837	15983,73	70792,01
5000	160,1754	705,9995	4125,845	27470,39	128960,3
10000	136,3645	1081,199	5176,091	27591,78	179030,7
25000	151,1798	968,9531	6051,531	34390,36	212097,9
35000	168,113	1049,968	8015,92	38346,41	242030,1
50000	189,3981	1163,605	7909,595	49775,11	240835
60000	166,9058	1316,127	8328,216	42092,12	268568
75000	203,5668	1209,468	6828,914	51401,5	273677,5
90000	231,1639	1528,534	8509,182	46901,08	296760,8
100000	218,8641	1185,224	7865,951	44587,66	344320,4

(3) Taula dels valors de ζ extrets a partir del número nodes visitats en les búsquedes del veí més proper

NodesVisitats \ k	2	3	4	5	6
1000	0,341733	0,471695	0,577188	0,668842	0,747348
5000	0,292155	0,404352	0,50143	0,580488	0,655097
10000	0,277346	0,386256	0,473985	0,557282	0,630703
25000	0,263316	0,361885	0,447141	0,526908	0,600499
35000	0,259715	0,353913	0,439675	0,516447	0,587576
50000	0,256535	0,342157	0,426878	0,507513	0,57664
60000	0,253256	0,340206	0,420539	0,500772	0,568816
75000	0,247890	0,335502	0,416436	0,490238	0,561602
90000	0,246566	0,331029	0,411939	0,485014	0,556307
100000	0,243696	0,329451	0,409385	0,481584	0,554776