

*A k legközelebbi szomszéd probléma
közelítő megoldása molekulák
fingerprintjein*

Fekete István (ELTE)

BME Matematikai Modellalkotás Szeminárium

2018. November 6.

Tartalom

- Bevezetés (előzmények)
- Molekulák kódolása
- Fingerprintek előállítása
- Hasonlósági keresés és a k - NN probléma
- Helyzetérzékeny hashelés (LSH)
- A k - NN program
- Véletlen bitsorozatok és az NN probléma
- Szabályosan tömörödő bitsorozatok és az LSH
- Valós fingerprint adatbázis és az LSH
- A keresés pontosságának javítása konvergencia figyeléssel

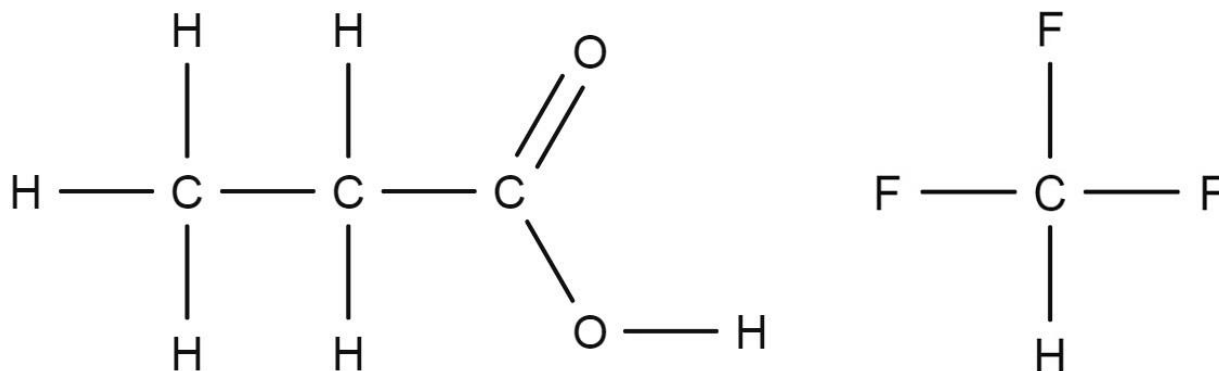
1. Bevezetés

Időbeli előzmények:

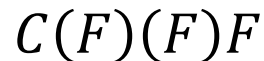
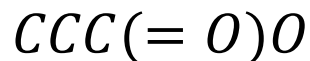
- 2010 elején két alkalmazott matematikus hallgató, *Kovács Balázs* és *Tamaga István*, TDK témát keresett.
- Témakiírás, ChemAxon Kft: „*Molekulagráfok leíróinak vizsgálata a hasonlósági keresés szempontjából*” témavezetők: *Kovács Péter* (ChemAxon), *Fekete István* (ELTE).
- A TDK dolgozat 2010-ben elkészült: Kari 1. díj, OTDK 3. díj (2011)
- TÁMOP 2010-12 pályázat egyik projektje: „*Keresési feladatok molekulagráfokon*” (Fekete I., Kovács P., Tichler K. és 8 hallgató).
- Elkészült: a fenti TDK dolgozat, hét szakdolgozat (2010-2012), később egy cikk és egy diplomamunka, majd egy PhD értekezés (egyik fele).

2. Molekulák kódolása

- Szerkezeti gráfok (propionsav: $C_3H_6O_2$ és fluoroform: CHF_3)



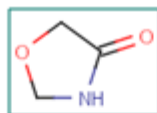
- *SMILES* kód:



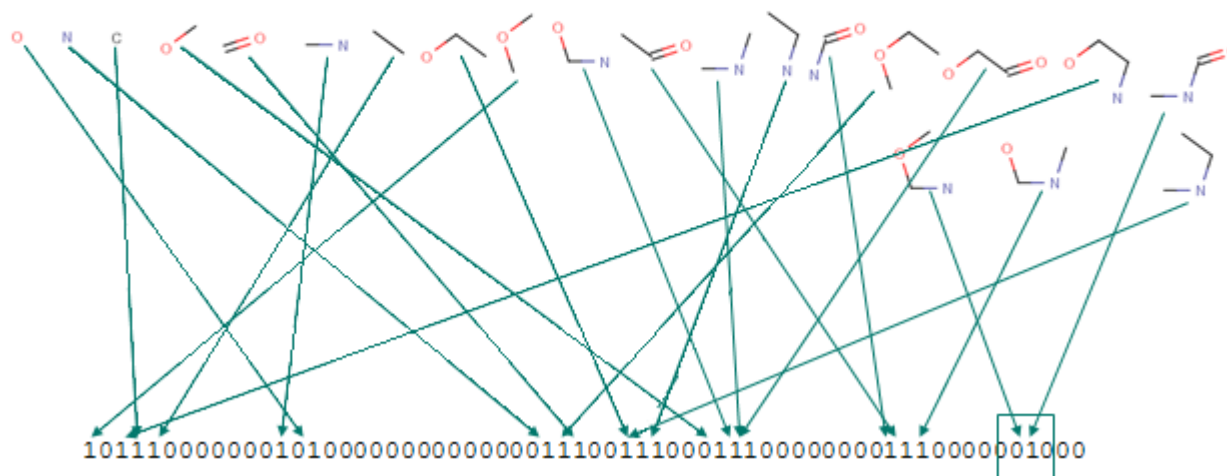
- A *SMILES* kód a molekulát reprezentáló gráf mélységi bejárásával (a *H* atomok elhagyásával) kapható. Ez a legtömörebb visszafejthető kódolás.

3. Fingerprintek előállítása

- Fingerprint: a molekulákat kódoló hosszú 0-1 sorozat.
- Az 1-es biteket a gráfbeli legf. s hosszú utak alapján állítjuk be (pl. $s=7$).
- A fingerprint megőrzi bizonyos szerkezeti tulajdonságokat.
- A fingerprintek összehasonlítása egyszerű és távolságuk közelítően tükrözi a kémiai hasonlóságot.



- Példa: *oxazolidin*, *PBFP 64-5-1*



4. Hasonlósági keresés és a k -NN probléma

- Gyakorlati alapfeladat:
 - Egy q lekérdező fingerprinthez megtalálni a k számú legközelebbi fingerprintet (a gyakorlatban $k = 8..20$).
 - Valós feladat paramétereit $n=236.617$, $d=1024$ (NCI adatbázis). Vannak több tízmilliós adatbázisok is.
- Leggyakoribb a távolságalapú megközelítés.
- *Hamming*-távolsággal: az eltérő értékű bitpozíciók száma:

$$\begin{array}{rcccccccccc} q = & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ x = & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 4 = d_H(q, x) \\ y = & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 6 = d_H(q, y) \end{array}$$

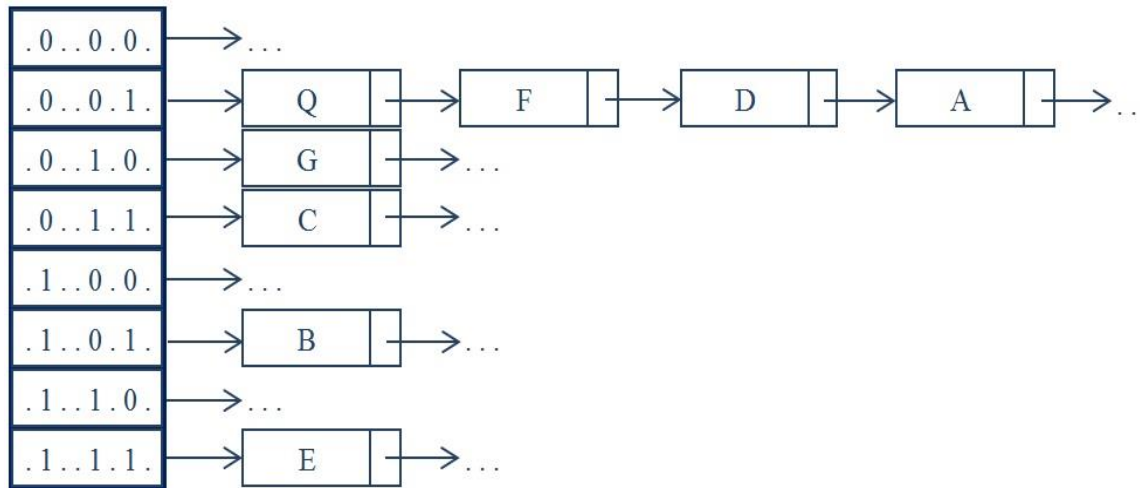
5. Helyzetérzékeny hashelés (*LSH*) (1/3)

- Hashelés: *FP*-ek elhelyezése r számú véletlen pozíció alapján 2^r vödörbe.
- A példában $n = 8$, $d = 8$ és $r = 3$, random pozíciók: 2, 5, 7.
- A k -*NN* keresés ($k=3$) pontos eredménye: *D*, *C*, *F*.
- Az *LSH* keresés közelítő eredménye: *D*, *A*, *F*. (Lásd: következő dia!)

Q	1	0	0	1	0	1	1	0	0
A	0	0	1	1	0	0	1	1	4
B	0	1	1	0	0	0	1	1	6
C	0	0	1	1	1	1	1	0	3
D	1	0	1	0	0	1	1	0	2
E	0	1	0	0	1	1	1	1	5
F	0	0	1	1	0	1	1	1	3
G	0	0	0	1	1	1	0	1	4

5. Helyzetérzékeny hashelés (*LSH*) (2/3)

- A példa folytatása: a véletlen módon választott 2, 5 és 7. pozíciók alapján történő hash-elés eredménye:



- A három ($k=3$), Q -hoz legközelebbi FP keresése kizárólag abban a vödörben, amelyik Q -t tartalmazza! (Hatékonyabb, de tévedhet.)
- Közelítő eredmény: F, D, A . Elvesztettük C -t, mivel az egyik hash-pozíció az egyik különbség-bitre esett.

5. Helyzetérzékeny hashelés (LSH) (3/3)

- Az LSH hibája (valós NCI adatok: $d = 1024$, $r = 10$, $\delta \approx 40$):
 - Egy közeli szomszéd megtalálásának a valószínűsége:
 $P_M = 0,6646$ (részletes becslés a dolgozatban!)
 - Egy közeli szomszéd elvesztésének a valószínűsége:
 $P_E = 1 - P_M = 0,3354$
- Az elveszített szomszéd visszanyerése több hasheléssel:
 - Kétszeres hashelés: $P_M^{(2)} = P_M + P_E P_M = 0,8875$
 - Háromszoros hash: $P_M^{(3)} = P_M (1 + P_E + P_E^2) = 0,9836$
 - Hash-sorozat határértéke: $\lim_{j \rightarrow \infty} P_M^{(j)} = \lim_{j \rightarrow \infty} P_M (1 + P_E + P_E^2 \dots + P_E^{j-1}) = P_M \cdot \frac{1}{1 - P_E} = 1$

6. A k -NN program

(1/2)

gyógyszer-molekula fingerprintje

CCOC(=O)c1c(O)c2ccc(C)c(Cl)c2n1

a nyolc legközelebbi hasonló gyógyszer-molekula

egzakt megoldás, keresés a teljes molekula-adatbázisban (250 ezer rekord)

közeli megoldás (random hash-elés): egy hibás találat, de a keresési tér 11,6%-ra csökkent

Rank	ID	Fingerprint
4	240445	001110001101111000011010
8	179361	001110001101111000011010
11	35908	001110001101111000011010
13	35190	001110001101111000011010
13	159613	001110001101111000011010
13	245978	001110001101111000011010
16	192638	001110001101111000011010
17	143513	001110001101111000011010
17	143513	001110001101111000011010

Rank	ID	Fingerprint
4	240445	001110001101111000011010
8	179361	001110001101111000011010
11	35908	001110001101111000011010
13	159613	001110001101111000011010
13	245978	001110001101111000011010
16	192638	001110001101111000011010
17	143513	001110001101111000011010
17	247477	001110001101111000011010

Min	Max	Avg
4	632	319,06

Min	Max	Avg
4	400	163,57

6. A k -NN program

(2/2)

Nearest Neighbor Search v2014.09.03

Generate input vectors
Number of vectors (n): 100000
Dimension (d): 1024
Display limit: 100
 Random display
 Random generation Cluster-based
Generate Save FPs

Cluster-based generation settings
Number of clusters: 64
Min. modified bits: 1
Max. modified bits: 5
 Center-based
 Mutation model

Load input vectors from text file
Display limit: 50
 Random display
Load

Hash table settings for Locality Sensitive Hashing
dim = 10, count = 1
Dimension: 10
Count: 1
Size: 1024
 Random coord. selection
 Use coordinate list
Add Del Clear Load

Input vectors
1 Add
68022. 11111001011101010010011111000
5468. 01001010110010010111010001101
46377. 01010011110011111000001101010
77244. 0101000010001001100011011100
43544. 00011101111001100110011011100
8838. 00011101111001100110011011100
63084. 11101000001110100110001100100
83383. 01101110100110010011101001100
57383. 01001100100110100101101000110
80573. 00000111000110110110011000001
90734. 01010110010111010110011100000
62718. 1111100101110101001001111000
73811. 11000100101101110000101101100
37901. 00011101000001010100110111111
97671. 00011111110000111110110111111
94259. 11111000101010000011001100111
43528. 10010011111001100100001111111
70907. 11111000101010000011001101011

Simple search - k-NN results
4 - 39687. 100100111110011001000011
5 - 30938. 100100111110011001000011
7 - 20364. 100100111110011001000011
7 - 33428. 100100111110011001000011
7 - 78546. 100100111110011001000011
8 - 85745. 100100111110011001000011
9 - 11724. 100100111110011001000011
10 - 40923. 100100111110011001000011
10 - 77137. 100100111110011001000011
11 - 5776. 100100111110011001000011
Search time: 00:00:00.1420
Dist. calculations: 99999 (100.00%)
Min. distance: 4
Max. distance: 11
Avg. distance: 7.80

LSH search - k-NN results
4 - 39687. 100100111110011001000011
5 - 30938. 100100111110011001000011
7 - 20364. 100100111110011001000011
7 - 33428. 100100111110011001000011
7 - 78546. 100100111110011001000011
8 - 85745. 100100111110011001000011
9 - 11724. 100100111110011001000011
10 - 40923. 100100111110011001000011
10 - 77137. 100100111110011001000011
11 - 5776. 100100111110011001000011
Search time: 00:00:00.0100
Dist. calculations: 1271 (1.27%)
Min. distance: 4
Max. distance: 11
Avg. distance: 7.80

Multi-search
Number of vectors to search:
 All vectors
 Sample size: 10000
Number of neighbors (k): 5
Search

Main parameters: n = 100000, d = 1024
Generation method: cluster-based, incr.
Generation time: 00:00:04.9680
Statistics

k-NN search
Number of neighbors (k): 10
Search

Simple search - Histogram of distances
Min: 4
Max: 553
Avg: 504.35
Export chart
Enlarge chart

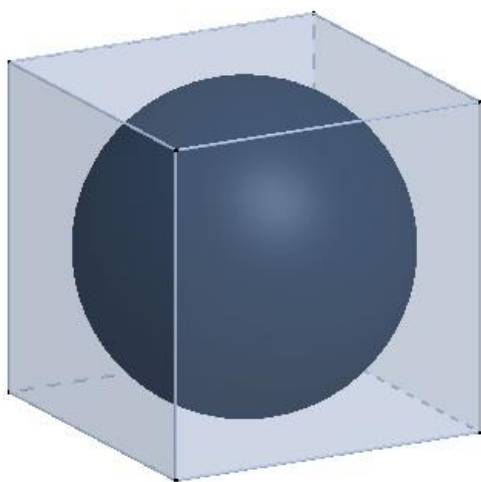
LSH search - Histogram of distances
Min: 4
Max: 498
Avg: 32.11
Export chart
Enlarge chart

7. Véletlen bitsorozatok és a legközelebbi szomszéd probléma (1/3)

- Generáljunk $n=236.617$, $d=1024$ hosszú véletlen 0-1 sorozatot
 - Ezek az „adatpontok” kevesen vannak 2^{1024} elemű térben.
 - Várhatóan 512 távolságra helyezkednek el egymástól, páronként.
 - A legközelebbi szomszéd bármelyik lehet, esetleges, hogy éppen melyik az \Rightarrow nincs értelme az NN problémának.
- Keressük annak valószínűségét, hogy legfeljebb l távolságra van q a p -től: $P(d_H(p, q) \leq l)$. Legyen $l=512-5 \cdot 16=432$.
- Binomiális eloszlás mellett:
 - $P(d_H(p, q) \leq 432) \leq \frac{1}{2^{1024}} \binom{1024}{432} \frac{593}{161} = 3.29 \cdot 10^{-7}$
- Normális eloszlás mellett, 5-szigma szabállyal:
 - $P(d_H(p, q) \leq 432) = 2.867 \cdot 10^{-7}$

7. Véletlen bitsorozatok és a legközelebbi szomszéd probléma (2/3)

- Dimenziószám növekedésével az adattér mérete exponenciálisan növekszik.
- Nagy adatmennyiség mellett is ritka kitöltést eredményez.
- Ez a jelenség a „*dimensionalitäts bosszúja*” (*curse of dimensionality*).



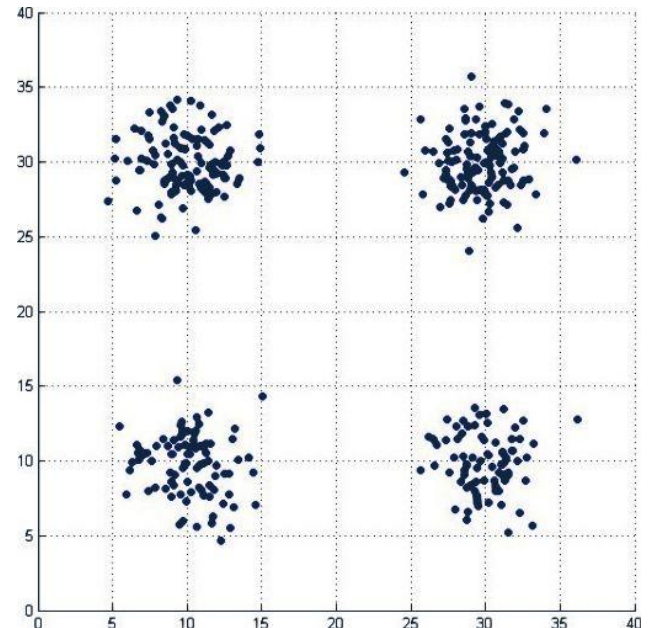
- Szemléltetése a kockába írt gömbbel. (A tér nagy része „távol van” a középponttól, a „sarkok felé” helyezkedik el.)
- A térfogatok aránya drasztikusan csökken
 - $d=3$ esetén 0.523599
 - $d=8$ esetén 0.015854
 - $d=14$ esetén 0.000037

7. Véletlen bitsorozatok és a legközelebbi szomszéd probléma (3/3)

- Az *LSH*-megoldás létezik ugyan, de esetleges:
 - Nincs gyakorlati haszna.
 - 4.5 szigma szabályra alapozva, annak valószínűsége, hogy két elem közti távolság legalább 446: $P(d_H(p, q) \leq 446) = 0,804$
 - Annak valószínűsége, hogy a keresett szomszéd egy másik edénybe kerül: $P = 1 - (578/1024)^{10} = 0,9967$
- Hasító pozíciók számának csökkentése:
 - A p és q fingerprintek két edény ($r=1$) esetén is közel 0,5 valószínűséggel kerülnek különböző edénybe.
- A program igazolja a számításokat:
 - Gyakorlatilag soha nem kapjuk vissza az egzakt eredményt.

8. Szabályosan tömörödő bitsorozatok és az *LSH* (1/3)

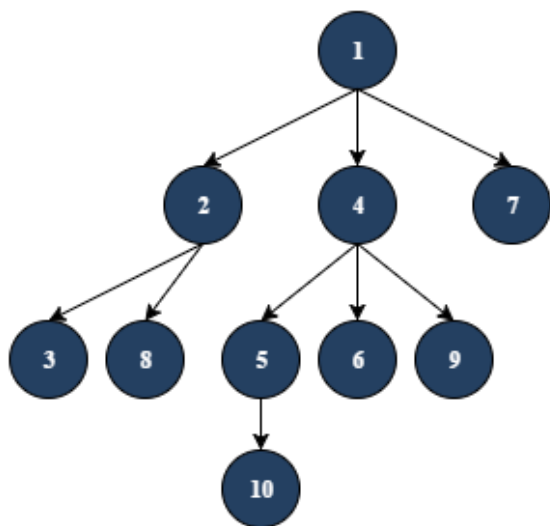
- Mesterségesen generált adatok másik „véglete”: szabályszerű klaszterek.
- Azt várjuk, hogy a helyzetérzékeny hasítás (*LSH*) magas találati aránnyal működik a szabályos klaszterek esetén.
- A generált klaszterek centrumait egyenletes eloszlás szerint vesszük fel.
- A klaszterek generálásának paraméterei:
 - n a bitsorozatok száma
 - d a bitsorozatok hossza
 - m a tömörödések száma
 - t_1, t_2 a távolságok alsó és felső korlátjai
- A klaszterek generálása több véletlen választás hierarchikus szekvenciája:
 - klaszter - szülő sorozat - t távolság ($t_1 \leq t \leq t_2$) - átbillentendő pozíciók.
- A középpontok várható távolsága $d/2$.



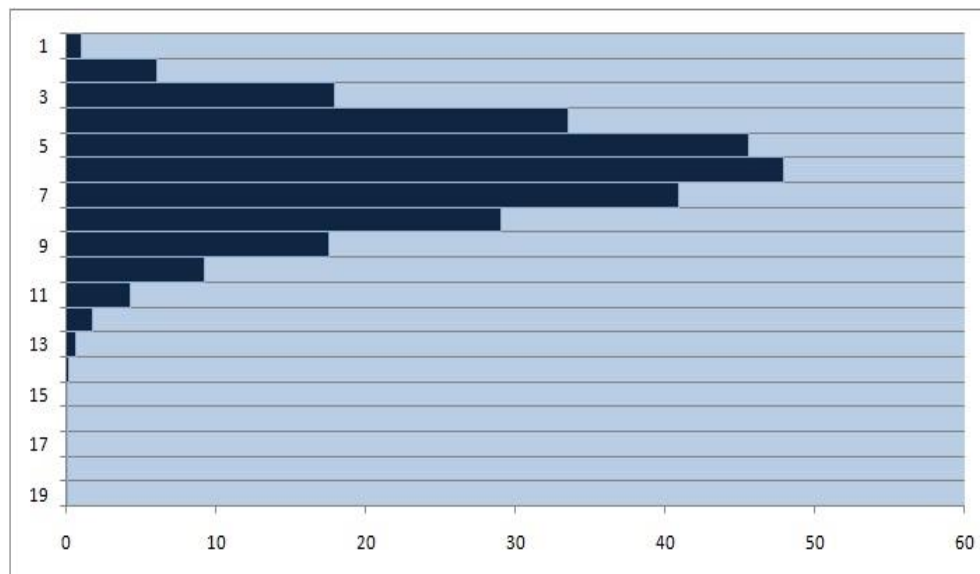
8. Szabályosan tömörödő bitsorozatok és az *LSH*

(2/3)

- Egy klaszter előállításának alapját a rekurzív fa képezi.
- A következő csúcs bármelyik előzőnek a gyereke lehet.



- A rekurzív fa „egyben marad, nem csúszik szét”. Számos szimuláció igazolja!



- Általánosítás: ha új csúcsot generálunk, akkor a szülő és az új gyerek csúcs közötti távolságot is sorsoljuk (adott határok között).

8. Szabályosan tömörödő bitsorozatok és az *LSH* (3/3)

- Az *LSH* a lekérdező q fingerprintnek mind a k számú közeli szomszédját megtalálja: a „0-s” oszlopban a valószínűségek.

- További oszlopokban a növekvő hibázások valószínűségei; a sorok egyre nagyobb szülőgyerek távolságoknak felelnek meg.

Távolság	0	1	2	3	4	5	6	7	8
1-1	92.15	5.05	0.77	0.45	0.25	0.26	0.23	0.24	0.33
1-2	89.23	6.48	1.22	0.60	0.45	0.36	0.43	0.48	0.51
1-3	86.95	7.45	1.59	0.81	0.55	0.55	0.53	0.55	0.81
1-4	84.75	8.47	1.97	0.97	0.61	0.61	0.76	0.64	1.10
1-5	81.87	9.81	2.47	1.26	0.79	0.78	0.73	0.96	1.25
1-10	70.60	13.77	4.61	2.15	1.64	1.46	1.48	1.82	2.44
1-20	54.03	18.75	7.34	3.81	2.96	2.63	2.54	3.34	4.59
1-30	40.68	20.75	10.10	5.86	4.01	3.75	3.82	4.87	6.16
1-40	31.29	20.73	11.73	7.15	5.13	4.81	4.78	5.98	8.41
5-10	57.69	21.07	7.02	3.22	1.92	1.77	1.80	2.21	3.30
10-40	18.30	21.01	15.30	9.70	6.99	5.67	5.81	7.22	9.99
20-40	10.94	18.12	16.53	11.54	8.31	7.06	6.79	8.69	12.02

9. Valós fingerprint adatbázis és az *LSH*

(1/2)

- A valós fingerprint adatbázis paraméterei:

$$n=236.617, d=1024, k=8, r=10$$

- Nagymennyiségű k -NN keresés:

- Az adathalmazt véletlenszerűen 24 db 1000 elemű fájlra tagoljuk.
- 5 db random állomány minden FP -jére 10 független k -NN keresés.
- Az 5 fájlra 1-1 átlag sort kapunk, majd ezeket is átlagoljuk.
- Egy fingerprintre végrehajtott egy k -NN keresés outputja, alatta az alsó sor: 10 keresés hibázási összesítése.

```
Fingerprint id: 69642, Exact search results: id 69630 dist
24, id 69640 dist 39, id 17309 dist 45, id 97332 dist 48, id
186037 dist 48, id 69649 dist 53, id 69652 dist 53, id 69644
dist 54
```

```
2 3 1 1 0 0 1 2 0
```

9. Valós fingerprint adatbázis és az *LSH* (2/2)

- A két táblázat leírása az előző dián szerepel.
- Az *LSH* keresés találati aránya viszonylag alacsony
 - Az A legfeljebb két hibával végződő keresések találati aránya nem teszik ki a keresések 45%-át sem.

0	1	2	3	4	5	6	7	8
22.53%	11.75%	8.82%	8.46%	7.76%	7.92%	8.35%	10.60%	13.81%
22.57%	10.64%	9.39%	8.08%	8.14%	8.35%	8.82%	10.03%	13.98%
22.70%	10.83%	9.46%	8.50%	8.28%	8.22%	8.77%	10.44%	12.80%
22.90%	11.63%	9.29%	8.61%	8.00%	7.74%	8.13%	10.23%	13.47%
22.21%	12.21%	9.50%	8.02%	7.38%	7.81%	8.39%	10.26%	14.22%

$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{5}$	$\bar{6}$	$\bar{7}$	$\bar{8}$
22.58%	11.41%	9.29 %	8.33%	7.91%	8.01%	8.49%	10.31%	13.66%

10. A keresés pontosságának javítása konvergencia figyelésével (1/5)

- Az Felhasználói szoftver esetén az egzakt megoldása (a teljes k -NN keresés) nem kerül feltüntetésre.
- Az LSH keresés közelítő eredményének pontossága önmagában nem becsülhető.
- Iteratív keresést végzünk az eredmény konvergenciájának figyelésével.
- A hatékonyság miatt az iterációk száma max. 5, a keresési tér a teljes tér max. 15%-a.

- Példa egy iterált k -NN keresésre:

az 5. lépésben már nincs javulás, a összes keresés a teljes tér 13,63 %-ára terjedt ki.

	1	2	3	4	5	6	7	8	0.00%
1	80	100	100	103	103	104	106	106	0.21%
2	80	80	87	89	100	100	103	103	1.53%
3	66	80	80	87	89	100	100	103	2.60%
4	66	80	80	87	89	99	100	100	4.60%
5	66	80	80	87	89	99	100	100	4.69%

10. A keresés pontosságának javítása konvergencia figyelésével (2/5)

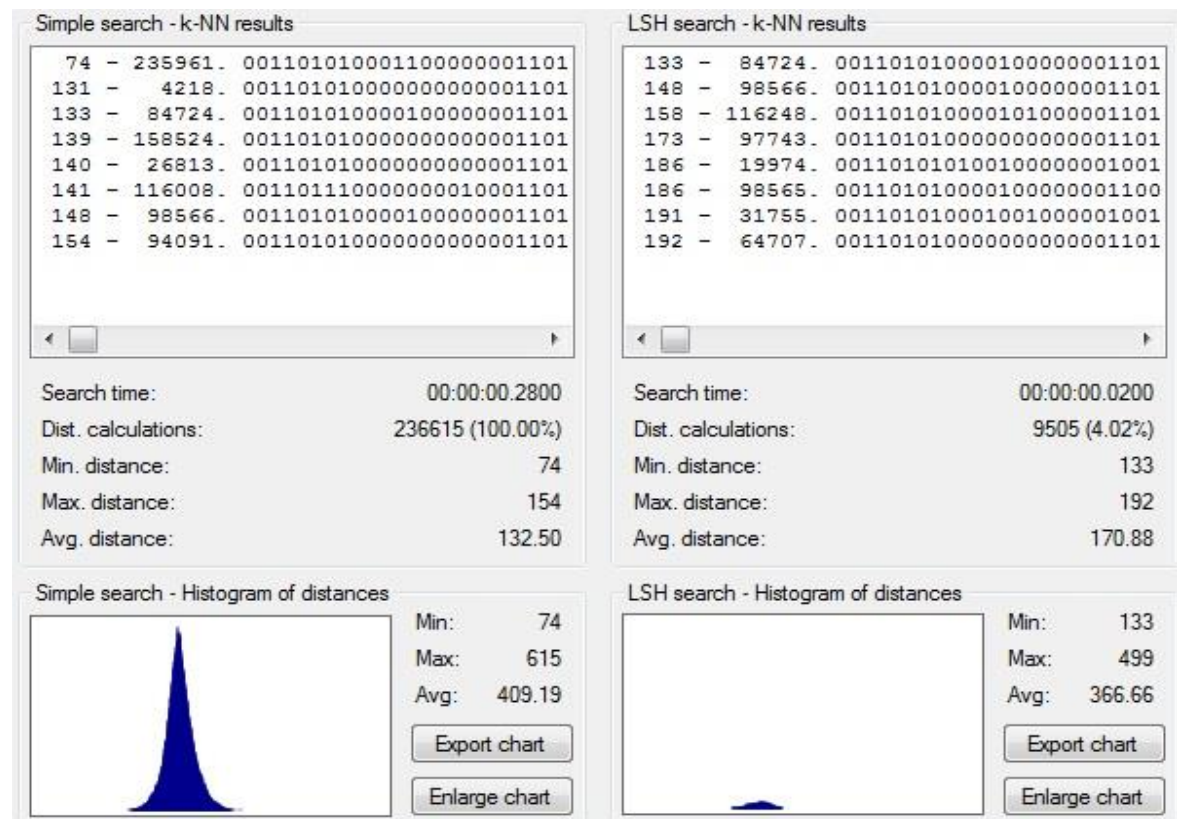
- Iteráció: egy közeli fingerprint elvesztésének valószínűsége csökken, ha az eredményt több hasítás uniója adja.
- A lekérdező q fingerprintet tartalmazó edények unióját képezzük, iterált módon, az újabb hashelések alapján.
- Példa: az *LSH* két iterációjával pontos eredményt kapunk: D, C, F (lásd: 6. dia).

Q	1	0	0	1	0	1	1	0
A	0	0	1	1	0	0	1	1
B	0	1	1	0	0	0	1	1
C	0	0	1	1	1	1	1	0
D	1	0	1	0	0	1	1	0
E	0	1	0	0	1	1	1	1
F	0	0	1	1	0	1	1	1
G	0	0	0	1	1	1	0	1

Q	1	0	0	1	0	1	1	0
A	0	0	1	1	0	0	1	1
B	0	1	1	0	0	0	1	1
C	0	0	1	1	1	1	1	0
D	1	0	1	0	0	1	1	0
E	0	1	0	0	1	1	1	1
F	0	0	1	1	0	1	1	1
G	0	0	0	1	1	1	0	1

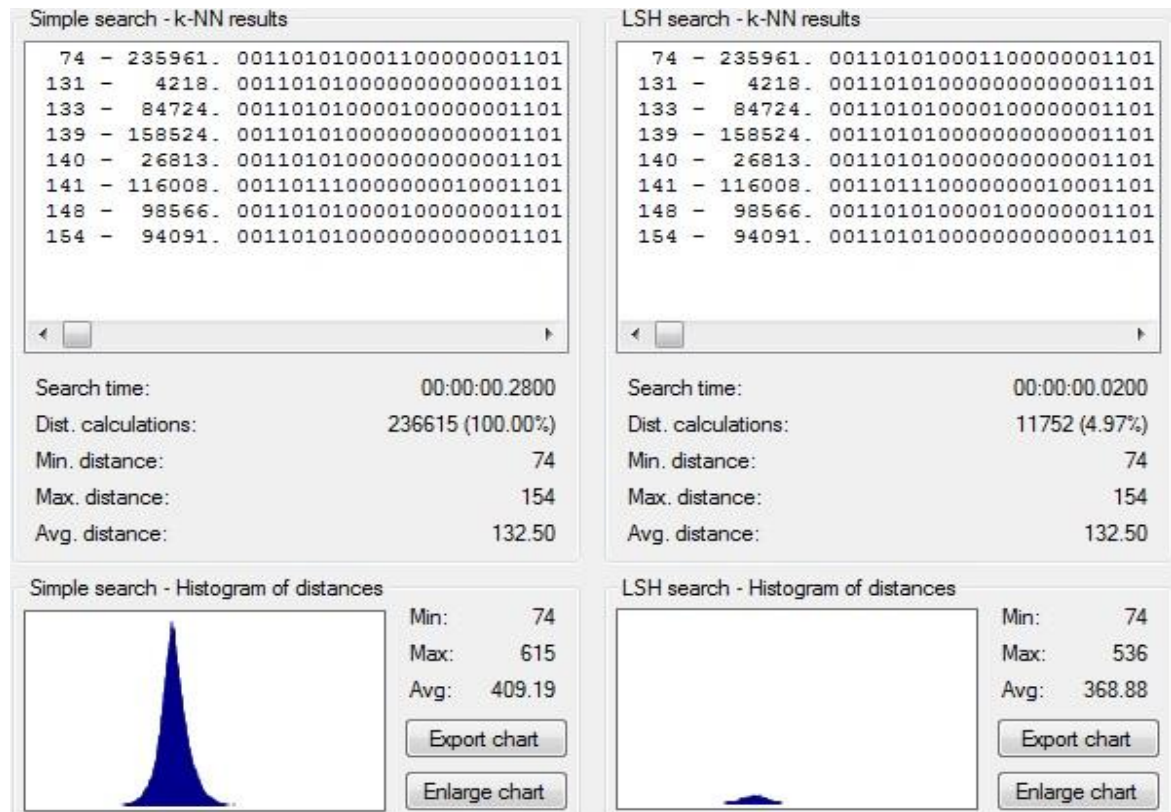
10. A keresés pontosságának javítása konvergencia figyelésével (3/5)

- Iterált *LSH* keresés: 1. lépés: 2 legközelebbi szomszédot talál meg.



10. A keresés pontosságának javítása konvergencia figyelésével (4/5)

- Iterált *LSH* keresés: 4. lépés: mind a 8 legközelebbi szomszédot megtalálja; a keresés tér korlátot is elérte közben.



10. A keresés pontosságának javítása konvergencia figyelésével (5/5)

- A várakozásnak megfelelően javul a hatékonyság:
A legfeljebb két hibával végződő keresések találati aránya 75% fölé emelkedett (vesd össze: 16. dia).

0	1	2	3	4	5	6	7	8
53.04%	15.98%	8.63%	5.63%	4.69%	3.82%	3.21%	2.54%	2.46%
55.10%	15.23%	8.04%	5.05%	4.50%	3.43%	3.28%	2.67%	2.70%
53.39%	14.76%	8.11%	6.00%	5.02%	3.90%	3.08%	2.67%	3.07%
52.81%	15.30%	8.21%	5.61%	4.35%	4.00%	3.24%	3.35%	3.13%
54.07%	15.04%	8.34%	5.81%	4.14%	3.56%	2.96%	3.10%	2.98%

$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{5}$	$\bar{6}$	$\bar{7}$	$\bar{8}$
53.68%	15.26%	8.27%	5.62%	4.54%	3.74%	3.15%	2.87%	2.87%