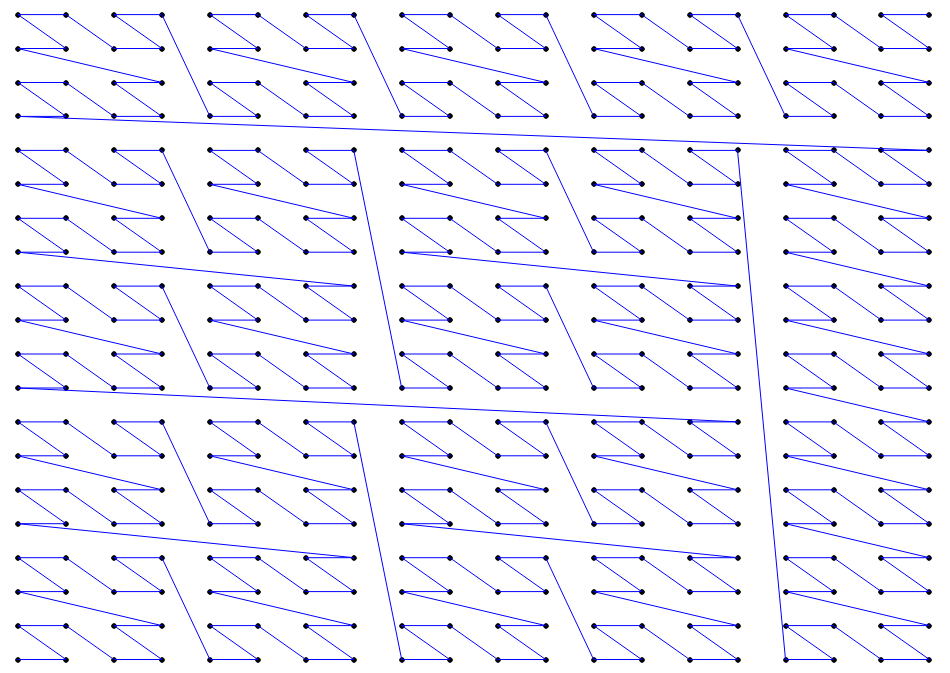
# Aproximační algoritmus

Algoritmus pro přibližné řešení používá Mortonovské uspořádání, které definuje pořadí n-dimenzionálních bodů v 1 dimenzi. Mortonovské uspořádání se normálně dělá tak, že se pro každou souřadnicovou komponentu dvou bodů prolnou bity v binárním formátu. Paper [1], podle kterého jsem algoritmus implementoval používá takzvaný Chanův trik [2] rozšířený funkčně i pro čísla s plovoucí desetinnou čárkou. Tento trik nám umožňuje porovnat dva n-dimenzionální body a srovnat tak celou množínu bodů pomocí paralelního quick sortu. Postupně tady používáme trik na všechny souřadnicové komponenty, poté podle určitého kritéria jednu dvojici vybereme a jejich porovnání předložíme jako vysledek porovnání n-dimenzionálních bodů.

Trik pro reálná čísla funguje tak, že pomyslně převedeme číslo do fixní reprezentace, tedy 2.75 bude 10.11b a použijeme Chanův trik pro celá čísla, který spočívá v nalezení nejvyššího bitu, kterým se dvě čísla liší. Ty souřadnicové komponenty, který mají tento bit nejvýše jsou vybrany pro výsledné porovnání.

Pokud se pokusíme uspořádat pravidelnou úplnou mřížku, jejíž body mají souřadnice tvaru (x \* 2n, y \* 2n, …), kde *n* je libovolné celé číslo a *x*, *y*, … = 0, 1, …, 2i pro různé body v této mřížce (*i* určuje velikost mřížky), dostaneme právě Lebesgueho prostor-vyplňující křivku. X a y nemusejí mít maximum v mocnině dvou, ale výsledná Lebesgueho křivka nebude kompletní.

Pozorování: Kód uvedený v paperu za těchto podmínek Lebesgueho prostor-vyplňující křivku nevydá, jen křivku, která se jí vzdáleně podobá. Upravil jsem proto tento kód tak, aby fungoval tak, jak naznačuje předchozí odstavec.



Poté, co body seřadíme, máme uspořádání, které hezky zachovává lokalitu, tedy pokud jdou body blízko po sobě v uspořádání, budou také blízko u sebe v prostoru. Pokud jsou blízko v prostoru, většinou také budou blízko v uspořádání, ale existují výjimky. V předcházející ilustraci jistě čtenář takové případy lehce najde.

Přibližné řešení pro nejbližší sousedy pro každý bod najdeme tak, vezmeme c \* k sousedů v uspořádání na obě strany a z nich vybereme k nebližších.

# Přesné řešení

Přesné řešení je založeno na vlastnosti Mortonovského uspořádání, že jednozančně definuje n-dimenzionální quadtree a že pokud křivka vzniklá tímto uspořádáním z nějakého uzlu tohoto stromu vyjde, pak se s ním už nikdy neprotne. Tento předpoklad funguje za podmínky, že zadané body tvoří úplnou pravidelnou mřížku. Jakmile některé z bodů v mřížce vynecháme nebo posuneme, může se stát, že křivka daná uspořádáním projde některým uzlem dvakrát. Autoři paperu však tuto vlastnost vyžadují pro správné fungování jejich přesného algoritmu, tudíž jeho správnost není garantována. Není vyloučeno, že by se podobný přístup dal stále využít za upravených předpokladů, bylo by to ale zřejmě složitější. Zároveň by také bylo nutné použít mou upravenou implementaci samotného Mortonova uspořádání.

V paperu je uveden link na repozitář autorského řešení, ten ale neexistuje, což mne utvrzuje v podezření, že algoritmus nefunguje tak dobře, jak autoři doufali.

# Vstup programu

Program očekává právě jeden textový soubor jako parametr. Tento soubor by měl obsahovat jedno přirozené číslo na první řádce. Toto číslo určuje K, tedy počet sousedů v KNNG. Všechny další řádky musí obsahovat buď dvě nebo tři reálná čísla oddělená mezerami určující souřadnice 2d nebo 3d vstupních bodů. Počet dimenzí bude automaticky vyvozen z tvaru řádek. Logicky není možné počet dimenzí měnit mezi jednotlivými řádkami souboru.

# Struktura programu

Program přečte vstupní soubor, najde minima všech souřadnic napříč celou množinou bodů, seřadí body podle Mortonova uspořádání s pomocí rozšířeného Chanova triku. To udělá za pomoci nalezených minim (toto je jedno z vylepšení, které jsem musel oproti v paperu navrhovanému algoritmu udělat, jelikož Mortonovo uspořádání nefunguje dobře se zápornými hodnotami). Dále pak projde uspořádané body a najde k nim hrany formující přibližné řešení. Hledání minimu, řazení a hledání přibližného řesení jsou paralelizované pomocí knihovny TBB.

# Reference

[1] M.Connor, P.Kumar. Fast construction of k-Nearest Neighbor Graphs for Point Clouds. In IEEE transactions on visualization and computer graphics, September 2009.

[2] T.M.Chan. Approximate nearest neighbor queries revisited. In SCG ’97: Proceedings of the thirteenth annual symposium on computational geometry, pages 352–358, New York, NY, USA, 1997.