Prostorová indexace s využitím gridu

Návod na cvičení, Geoinformatika.

Na vstupu je mračno bodů $P = \{p_i\}_{i=1}^{n}$, kde $p_i = [x_i, y_i, z_i]$. Nad mračnem zkonstruujeme pomocnou indexační strukturu reprezentovanou 3D gridem. Grid je tvořen jednotlivými buňkami (voxely), celkový počet buněk je funkcí velikosti datasetu n a prostorové dimenze, $k = 3$ volíme ho jako

$$n_b = n^{1/k} = n^{1/3}.$$

Počet buněk n_r resp. n_c v řádku resp. v sloupci gridu je roven

$$n_r = n_c = n_b^{1/k} = n_b^{1/3}.$$

Velikost buňky. Velikost buňky gridu určíme ze vztahu

$$b_x = \frac{\overline{x} - \underline{x}}{n_r}, \qquad b_y = \frac{\overline{y} - \underline{y}}{n_r}, \qquad b_z = \frac{\overline{z} - \underline{z}}{n_r},$$

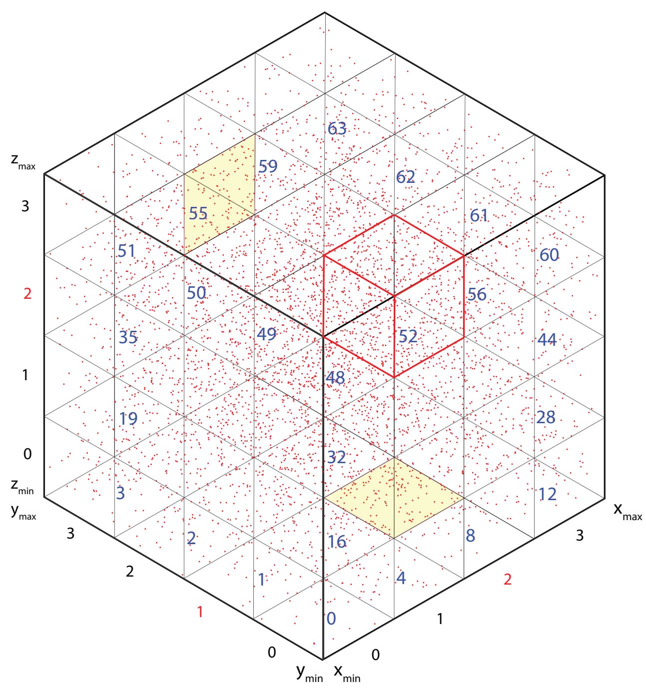
kde

$$\underline{x} = \min_{1\leq i\leq n}\{x_i\}, \qquad \overline{x} = \max_{1\leq i\leq n}\{x_i\}, \qquad \underline{y} = \min_{1\leq i\leq n}\{y_i\}, \qquad \overline{y} = \max_{1\leq i\leq n}\{y_i\}, \qquad \underline{z} = \min_{1\leq i\leq n}\{z_i\}, \qquad \overline{z} = \max_{1\leq i\leq n}\{z_i\}.$$

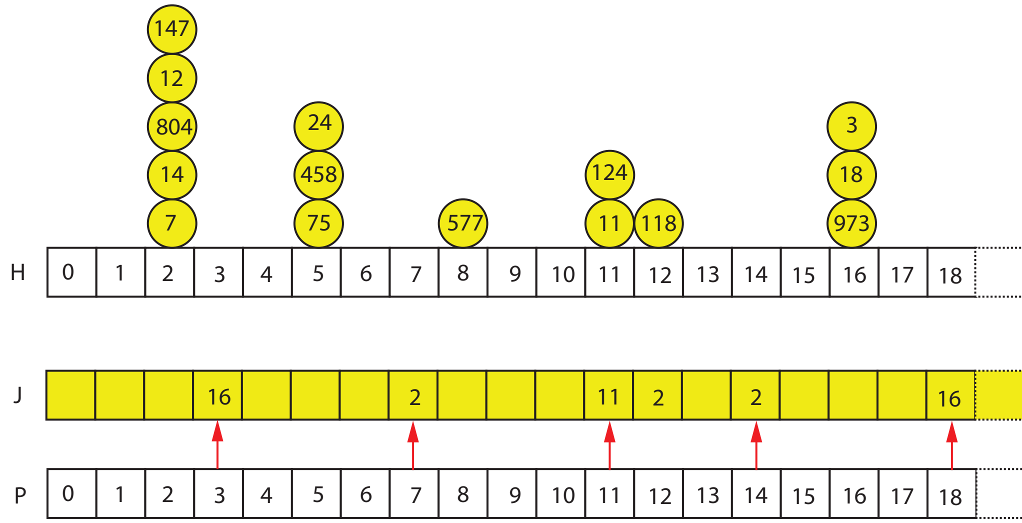
Normalizace souřadnic. Normalizované (redukované) souřadnice bodu $p_i = [x'_i, y'_i, z'_i]$ mají tvar

$$x'_i = \frac{x_i - \underline{x}}{\overline{x} - \underline{x}}, \qquad y'_i = \frac{y_i - \underline{y}}{\overline{y} - \underline{y}}, \qquad z'_i = \frac{z_i - \underline{z}}{\overline{z} - \underline{z}},$$

kde $0 \leq x'_i \leq 1$, $0 \leq y'_i \leq 1$, $0 \leq z'_i \leq 1$. Následně budou použity k výpočtu hashovací funkce.



Obrázek 1: Ukázka 3D indexačního gridu, znázorněny 3D a 1D indexy.



Obrázek 2: Ukázka prostorové indexace bodů množiny P, hashovací tabulka H a 1D pole J.

Výpočet indexů bodu. Poloha každé z buněk v gridu je určena trojicí indexů $\langle j_x, j_y, j_z \rangle$. Bod p_i leží v buňce gridu s indexy

$$j_x = \lfloor cn_r x'_i \rfloor, \qquad j_y = \lfloor cn_r y'_i \rfloor, \qquad j_z = \lfloor cn_r z'_i \rfloor,$$

kde $c = 0.99$ je "zaokrouhlovací" konstanta, symbol $\lfloor \cdot \rfloor$ představuje zaokrouhlení zdola (např. funkce `ceil()` či `int()`).

Hashovací funkce. Pro konverzi $\langle j_x, j_y, j_z \rangle$ na jednodimenzionální index j použijeme jednoduchou hashovací funkci

$$a = h(k),$$

kde

$$h(k) = j_x + j_y n_r + j_z n_r^2, \qquad a \equiv j, \qquad k \equiv p,$$

$h(k)$ je kvadratickou funkcí n_r. Číslování buněk probíhá po řádcích v jednotlivých vrstvách, každá z buněk má unikátní hash reprezentovaný indexem j, kde $0 \leq j \leq n_b - 1$, viz Obr. 1. Protože prostor klíčů je "větší" než prostor adres, dochází ke kolizím hashovací funkce, kdy uvnitř jedné buňky může být více bodů. Každá z n_b buněk gridu bude obsahovat průměrně

$$m = \frac{n}{n_b} = n^{2/3},$$

bodů p_i množiny P.

Datové struktury. Pro vlastní indexaci budou vytvořeny dvě pomocné datové struktury. První představuje *hashovací* tabulku, která pro každou buňku ukládá indexy j_i všech bodů, které jsou v ní obsaženy. Pro reprezentaci v programovacím jazyce Python bude použít Dictionary

```
H = {j1 : [i11, i12, ..., i1k1], j2 : [i21, i22, ..., i2k2], ...., jnb :[inb1, inb2, ..., inbk]}
```

tvořený n_b položkami. Druhou strukturu bude tvořit 1D pole s n položkami, které každému bodu p_i přiřadí jednodimenzionální index j_i

```
J = [j1, j2, ..., jn].
```

Dojde tak k obousměrnému prolinkování a budeme vědět, které body p_i se nachází v konkrétní buňce gridu, a ve které buňce gridu se nachází konkrétní bod p_i, viz Obr. 2.

Aplikace hashovací funkce. Pro libovolný "query point" $q = [x, y, z]$ postupujeme následujícím způsobem. S využitím hashovací funkce spočteme 1D adresu buňky j, která tento bod obsahuje. Dotazem do hashovací struktury

```
Q = H[j]
```

získáme podmnožinu s průměrnou velikostí m bodů, která je v buňce obsažena. Tyto body můžeme použít např. k rychlému nalezení nejbližšího souseda.

Zhodnocení efektivity. V každé z n_b buněk gridu se nachází průměrně $m = n^{2/3}$ bodů původní množiny, které při hledání nejbližšího souseda musíme prohledat. Tento krok musíme zopakovat postupně pro všechny buňky gridu, budeme tedy provádět $O(m \cdot n_b) = O(n)$ operací. Bez prostorové indexace budeme každému z n bodů prohledávat jeho n sousedů, celkově tedy budeme provádět $O(n \cdot n) = O(n^2)$ operací. Zatímco původní metoda má kvadratickou složitost, akcelerovaná metoda má pouze lineární složitost.