UI zadanie 2a

Vašou úlohou je naprogramovať klasifikátor pre nové body – v podobe funkcie classify(int X, int Y, int k), ktorá klasifikuje nový bod so súradnicami X a Y, pridá tento bod do nášho 2D priestoru (s farbou podľa klasifikácie) a vráti triedu, ktorú pridelila pre tento bod. Na klasifikáciu použite k-NN algoritmus, pričom k môže byť 1, 3, 7 alebo 15.

Vybral som sa smerom využitia KD tree, keďže boli ostatné metódy pre mňa neefektívne. Vyskúšal som aj rozdeliť si plochu na menšie štvorce ale tam som mal problémy.

Po konzultácií som dospel do bodu, že som si KD tree musel implementovať vlastnou pomocou bez použitia knižnice.

ŠTRUKTÚRA STROMU:

```
# Define k-d tree node
class KDTreeNode:
    def __init__(self, point, axis, left=None, right=None):
        self.point = point
        self.axis = axis
        self.left = left
        self.right = right
```

VYHĽADÁVANIE NAJBLIŽŠÍCH K SUSEDOV

```
def nearest_neighbors(self, target, k):
        best = []

        def _search(root):
            if root is None:
                return

            dist = euclidean_distance(target, root.point)
            if len(best) < k:
                heapq.heappush(best, (-dist, root.point))
            elif dist < -best[0][0]:
                heapq.heappushpop(best, (-dist, root.point))

            axis = root.axis
            diff = (target.x, target.y)[axis] - (root.point.x,
root.point.y)[axis]
            close, away = (root.left, root.right) if diff < 0 else
(root.right, root.left)

            _search(close)
            if len(best) < k or abs(diff) < -best[0][0]:
                _search(away)

        _search(self.root)
        return [point.label for _, point in sorted(best, reverse=True)]

def euclidean_distance(point1, point2):
    return math.sqrt((point1.x - point2.x) ** 2 + (point1.y - point2.y) ** 2)
```

Pri spustení funkcie `nearest_neighbors` je prijatý cieľový bod `target` a počet susedov `k`. Vyhľadávanie využíva vnorenú rekurzívnu funkciu `_search`, ktorá prechádza KD-Tree, pričom uprednostňuje bližšie podstromy. Funkcia `_search` používa prioritný zásobník (max heap) `best`, ktorý neustále udržuje len k najbližších bodov k cieľovému bodu.

KLASIFIKACIA

```
# Classification function
def classify(x, y, k, kdtree_root, points):
    neighbors = knn_search(kdtree_root, [x, y], k)
    nearest_classes = [neighbor[1]["class"] for neighbor in neighbors]
    most_common_class = Counter(nearest_classes).most_common(1)[0][0]
    points.append({"coords": [x, y], "class": most_common_class})  # Add to
points list but not initial_points
    kdtree_root = build_kdtree(points)  # Rebuild tree after each addition
    return most_common_class, kdtree_root
```

Táto funkcia je alfa omega kódu, kedy v podstate hľadá K najbižších susedov, podľa ktorých určuje triedu do ktorej bod klasifikuje, na nájednie najbližších bodov sa používa eklidova veta na výpočet vzdialenosti

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}.$$

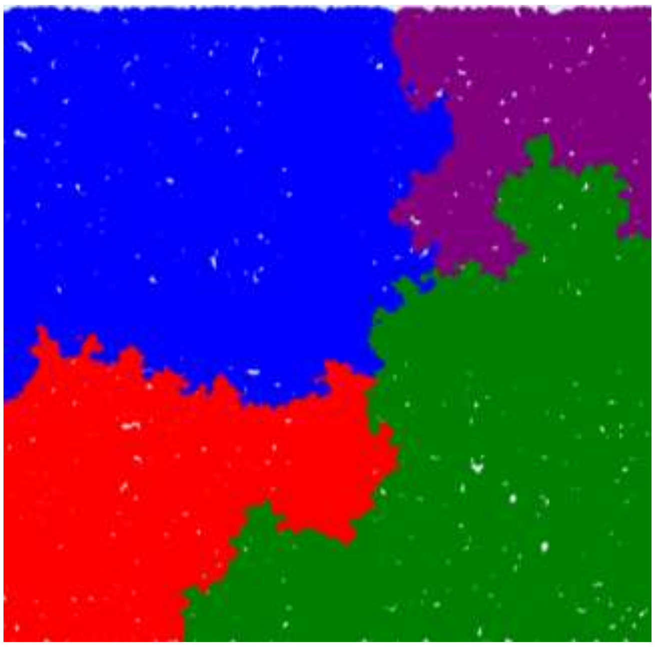
GENEROVANIE BODOV:

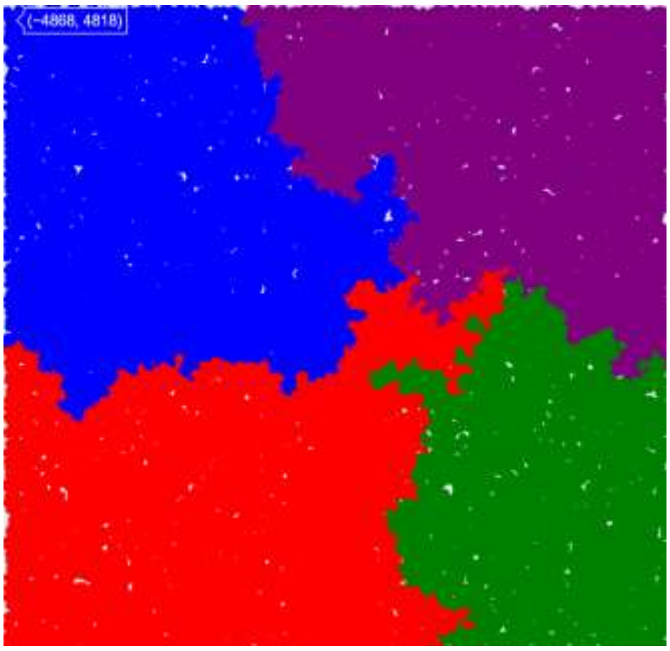
```

def generate_point(label):
    if label == 'R':
        if random.random() < 0.99:
            X = random.randint(-5000, 500)
            Y = random.randint(-5000, 500)
        else:
            X = random.randint(-5000, 5000)
            Y = random.randint(-5000, 5000)
```

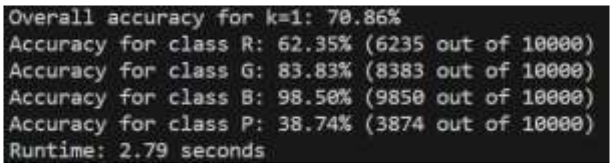
Generujeme body podľa podmienok, keď naskytne 1% ktoré uloží bod náhodne po mape, aj tak mu musíme prideliť pred klasifikáciou triedu,

EXPERIMENTY:

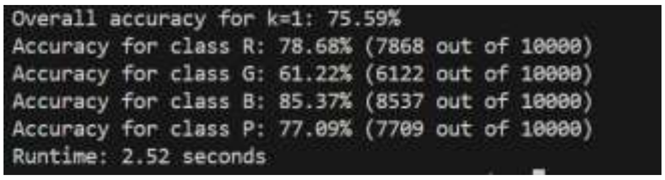
K=1, počas experimentu som si všimol, že vždy jedna trieda excelovala a zvyšné zaostávali



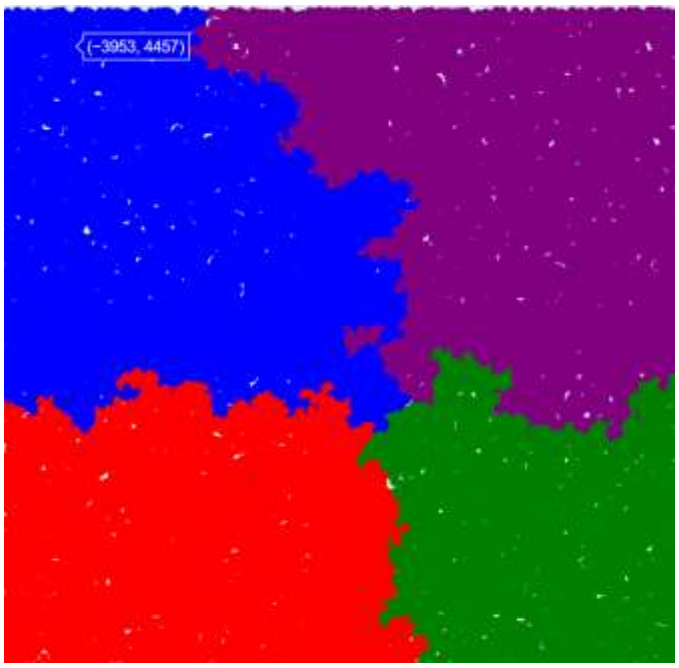


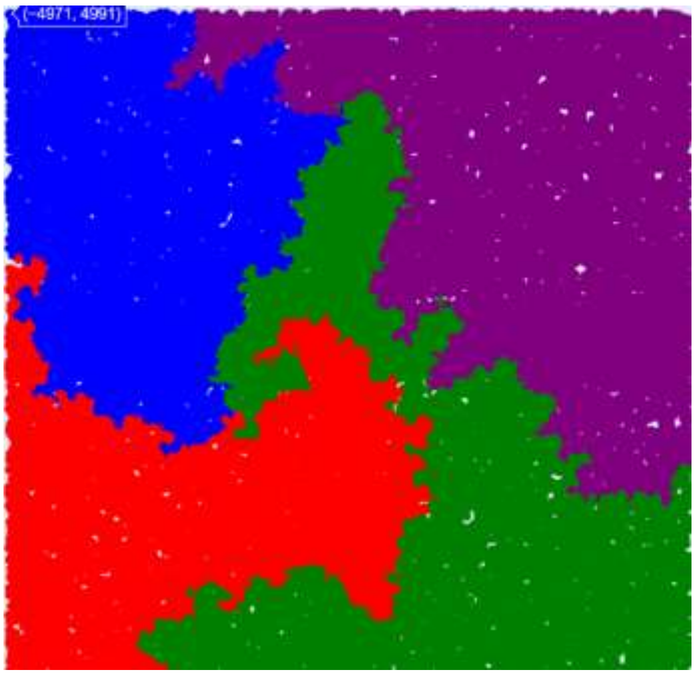


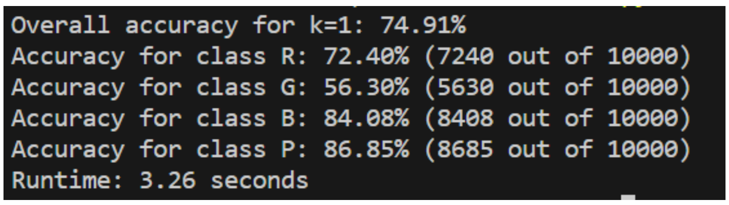
```
Overall accuracy for k=1: 70.86%
Accuracy for class R: 62.35% (6235 out of 10000)
Accuracy for class G: 83.83% (8383 out of 10000)
Accuracy for class B: 98.50% (9850 out of 10000)
Accuracy for class P: 38.74% (3874 out of 10000)
Runtime: 2.79 seconds
```



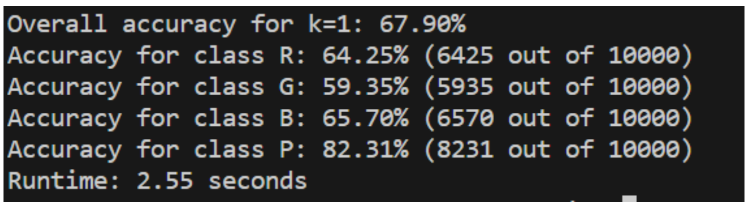
```
Overall accuracy for k=1: 75.59%
Accuracy for class R: 78.68% (7868 out of 10000)
Accuracy for class G: 61.22% (6122 out of 10000)
Accuracy for class B: 85.37% (8537 out of 10000)
Accuracy for class P: 77.09% (7709 out of 10000)
Runtime: 2.52 seconds
```





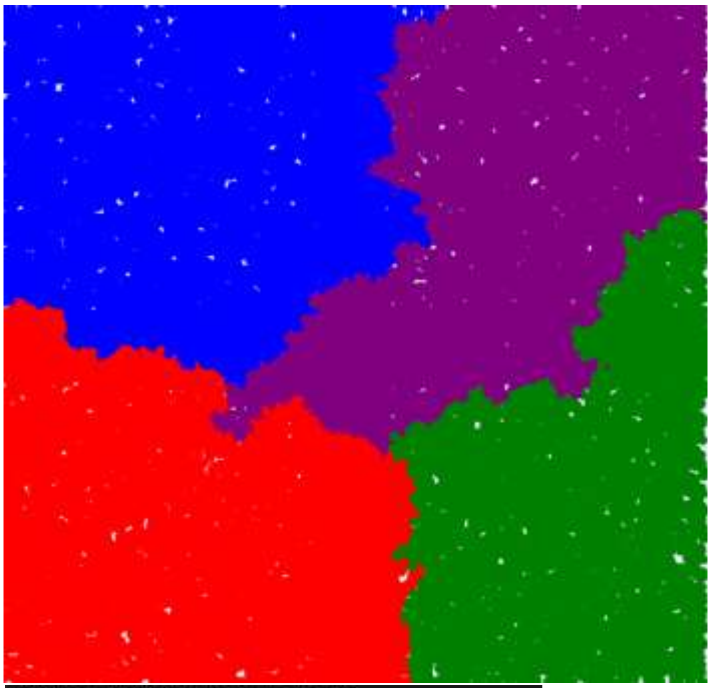


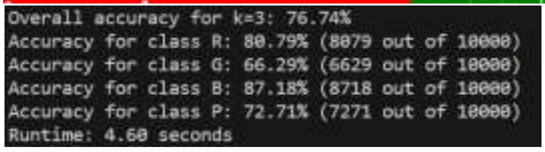
```
Overall accuracy for k=1: 74.91%
Accuracy for class R: 72.40% (7240 out of 10000)
Accuracy for class G: 56.30% (5630 out of 10000)
Accuracy for class B: 84.08% (8408 out of 10000)
Accuracy for class P: 86.85% (8685 out of 10000)
Runtime: 3.26 seconds
```



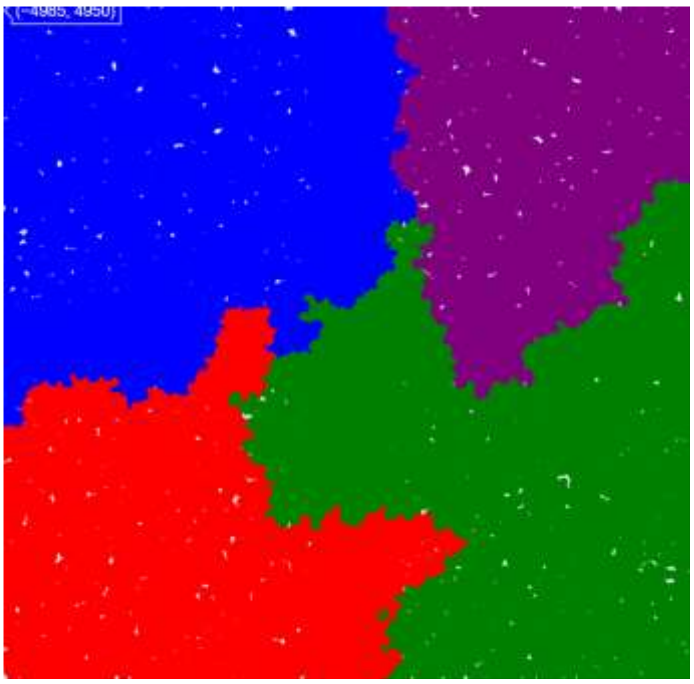
```
Overall accuracy for k=1: 67.90%
Accuracy for class R: 64.25% (6425 out of 10000)
Accuracy for class G: 59.35% (5935 out of 10000)
Accuracy for class B: 65.70% (6570 out of 10000)
Accuracy for class P: 82.31% (8231 out of 10000)
Runtime: 2.55 seconds
```

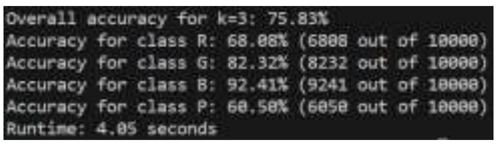
K=3



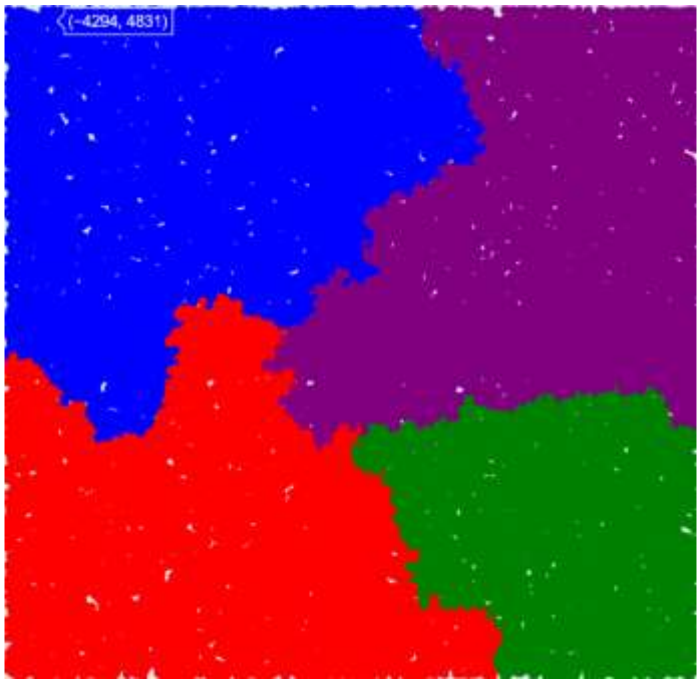


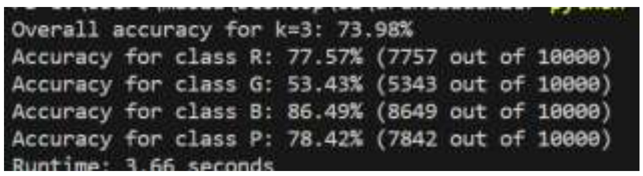
```
Overall accuracy for k=3: 76.74%
Accuracy for class R: 80.79% (8079 out of 10000)
Accuracy for class G: 66.29% (6629 out of 10000)
Accuracy for class B: 87.18% (8718 out of 10000)
Accuracy for class P: 72.71% (7271 out of 10000)
Runtime: 4.60 seconds
```



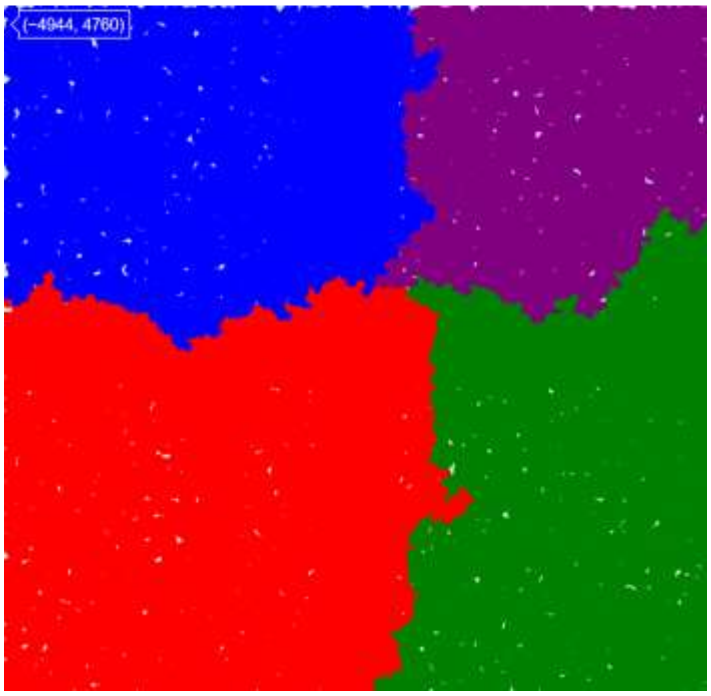


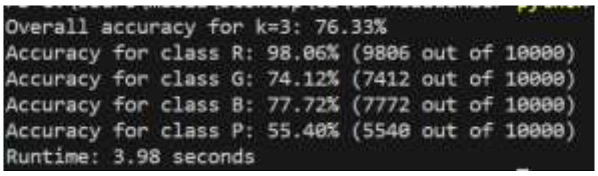
```
Overall accuracy for k=3: 75.83%
Accuracy for class R: 68.08% (6808 out of 10000)
Accuracy for class G: 82.32% (8232 out of 10000)
Accuracy for class B: 92.41% (9241 out of 10000)
Accuracy for class P: 60.50% (6050 out of 10000)
Runtime: 4.05 seconds
```





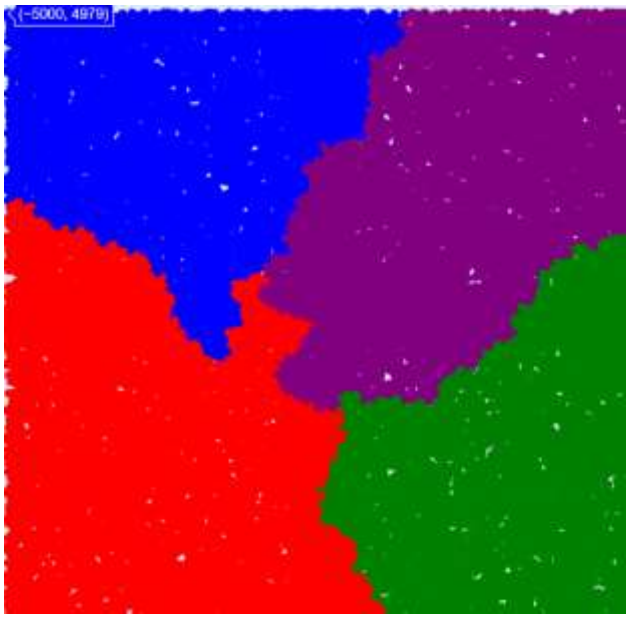
```
Overall accuracy for k=3: 73.98%
Accuracy for class R: 77.57% (7757 out of 10000)
Accuracy for class G: 53.43% (5343 out of 10000)
Accuracy for class B: 86.49% (8649 out of 10000)
Accuracy for class P: 78.42% (7842 out of 10000)
Runtime: 3.66 seconds
```



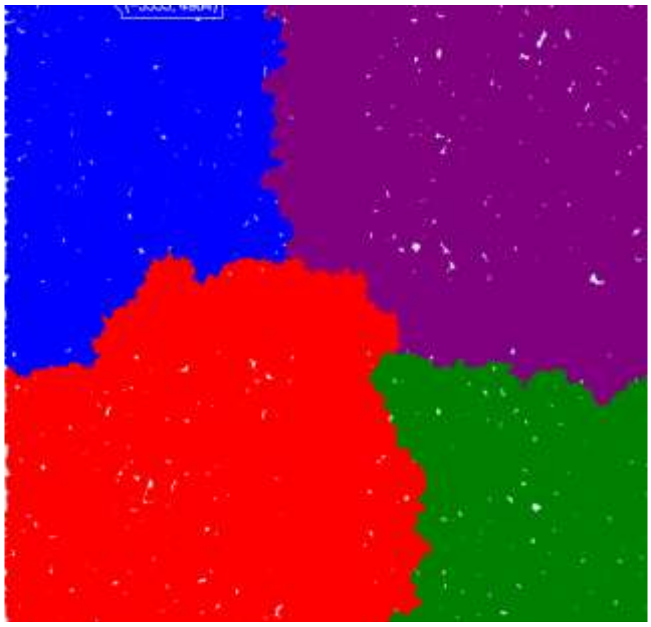


```
Overall accuracy for k=3: 76.33%
Accuracy for class R: 98.06% (9806 out of 10000)
Accuracy for class G: 74.12% (7412 out of 10000)
Accuracy for class B: 77.72% (7772 out of 10000)
Accuracy for class P: 55.40% (5540 out of 10000)
Runtime: 3.98 seconds
```

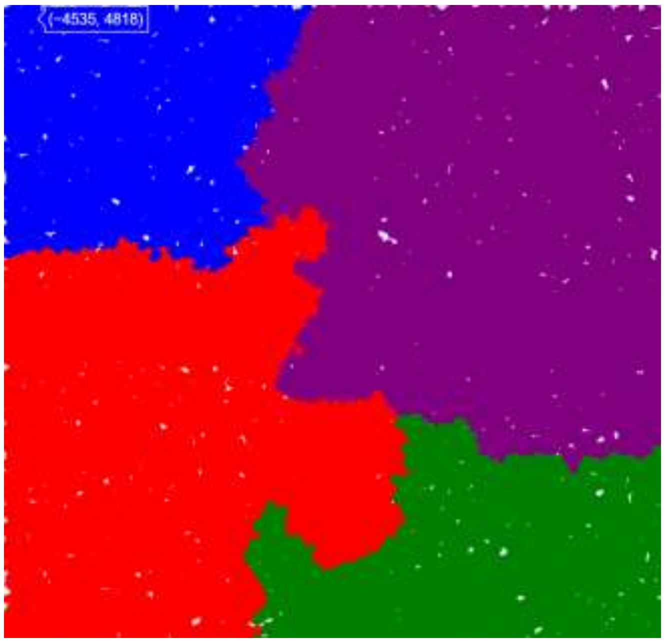
K = 7



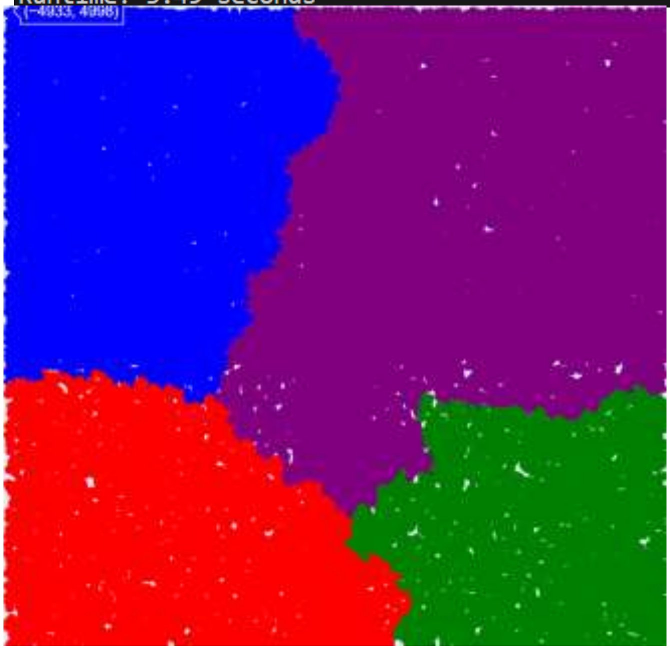
```
Overall accuracy for k=7: 76.00%
Accuracy for class R: 87.16% (8716 out of 10000)
Accuracy for class G: 68.69% (6869 out of 10000)
Accuracy for class B: 69.68% (6968 out of 10000)
Accuracy for class P: 78.47% (7847 out of 10000)
Runtime: 5.42 seconds
```



```
Overall accuracy for k=7: 75.99%
Accuracy for class R: 92.77% (9277 out of 10000)
Accuracy for class G: 50.85% (5085 out of 10000)
Accuracy for class B: 66.06% (6606 out of 10000)
Accuracy for class P: 94.28% (9428 out of 10000)
Runtime: 3.49 seconds
```

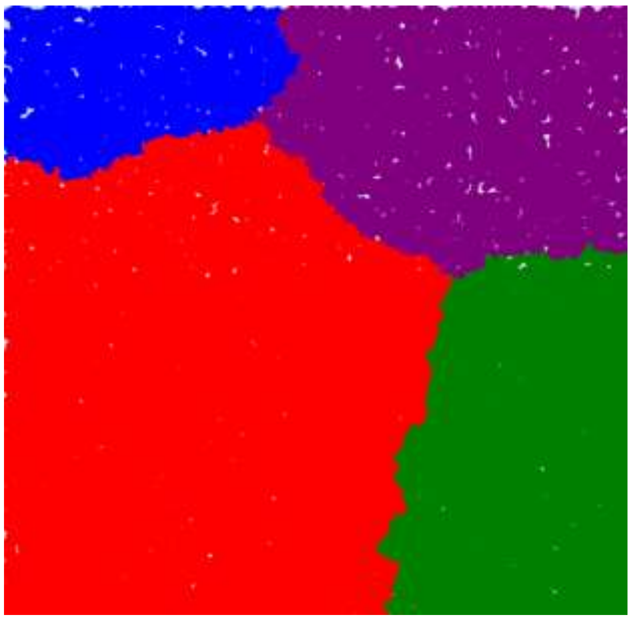


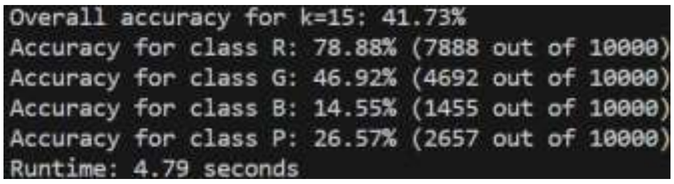
```
Overall accuracy for k=7: 70.41%
Accuracy for class R: 84.45% (8445 out of 10000)
Accuracy for class G: 48.44% (4844 out of 10000)
Accuracy for class B: 50.57% (5057 out of 10000)
Accuracy for class P: 98.19% (9819 out of 10000)
Runtime: 3.06 seconds
```



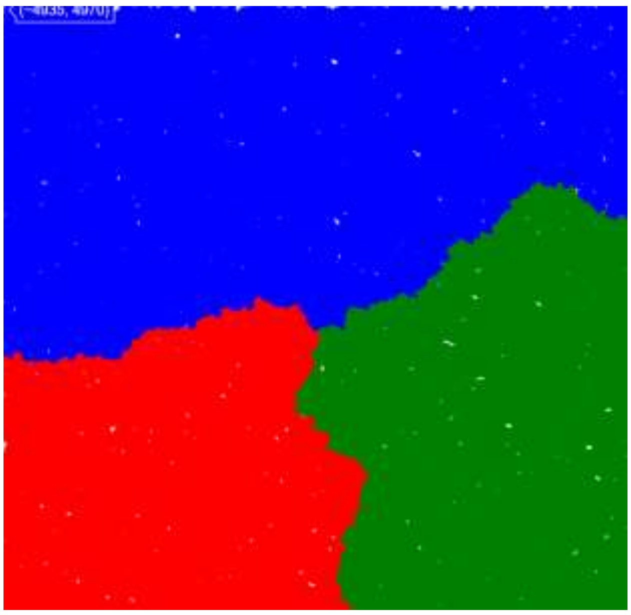
```
Overall accuracy for k=7: 64.50%
Accuracy for class R: 46.95% (4695 out of 10000)
Accuracy for class G: 37.44% (3744 out of 10000)
Accuracy for class B: 76.02% (7602 out of 10000)
Accuracy for class P: 97.60% (9760 out of 10000)
Runtime: 3.27 seconds
```

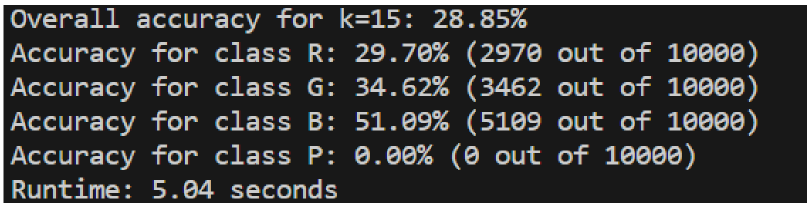
K = 15



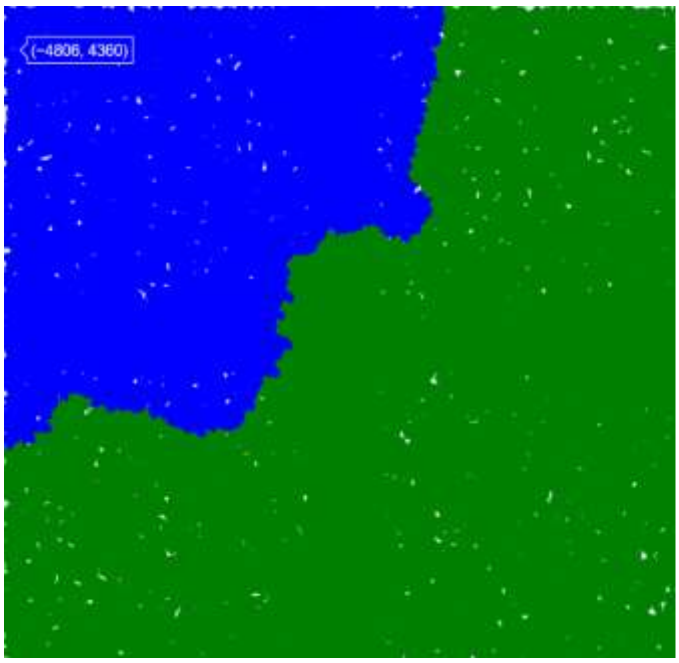


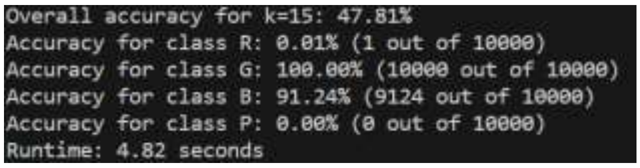
```
Overall accuracy for k=15: 41.73%
Accuracy for class R: 78.88% (7888 out of 10000)
Accuracy for class G: 46.92% (4692 out of 10000)
Accuracy for class B: 14.55% (1455 out of 10000)
Accuracy for class P: 26.57% (2657 out of 10000)
Runtime: 4.79 seconds
```



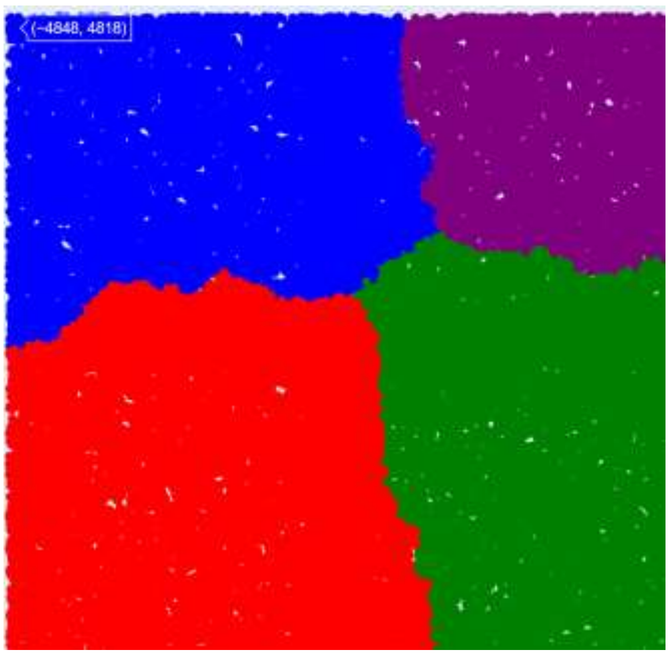


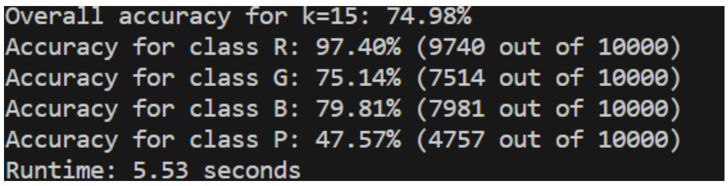
```
Overall accuracy for k=15: 28.85%
Accuracy for class R: 29.70% (2970 out of 10000)
Accuracy for class G: 34.62% (3462 out of 10000)
Accuracy for class B: 51.09% (5109 out of 10000)
Accuracy for class P: 0.00% (0 out of 10000)
Runtime: 5.04 seconds
```





```
Overall accuracy for k=15: 47.81%
Accuracy for class R: 0.01% (1 out of 10000)
Accuracy for class G: 100.00% (10000 out of 10000)
Accuracy for class B: 91.24% (9124 out of 10000)
Accuracy for class P: 0.00% (0 out of 10000)
Runtime: 4.82 seconds
```





```
Overall accuracy for k=15: 74.98%
Accuracy for class R: 97.40% (9740 out of 10000)
Accuracy for class G: 75.14% (7514 out of 10000)
Accuracy for class B: 79.81% (7981 out of 10000)
Accuracy for class P: 47.57% (4757 out of 10000)
Runtime: 5.53 seconds
```

Záver: Algoritmus na klasifikáciu považujem za funkčný. Je to zaujímavý algoritmus na experimentovanie, hlavne na K hodnotu. Skúšal som ešte experimentovať s hodnotamy spawnovania bodov. Napr:

K=20 spawn uplne nahodne