K近邻算法（KNN）

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
```

```
In [2]: np.random.seed(1)
        X1 = np.random.randint(1, 10, size=10).reshape(-1, 2)
        X1
```

```
Out[2]: array([[6, 9],
               [6, 1],
               [1, 2],
               [8, 7],
               [3, 5]])
```

```
In [3]: np.random.seed(1)
        X2 = np.random.randint(10, 20, size=10).reshape(-1, 2)
        X2
```

```
Out[3]: array([[15, 18],
               [19, 15],
               [10, 10],
               [11, 17],
               [16, 19]])
```

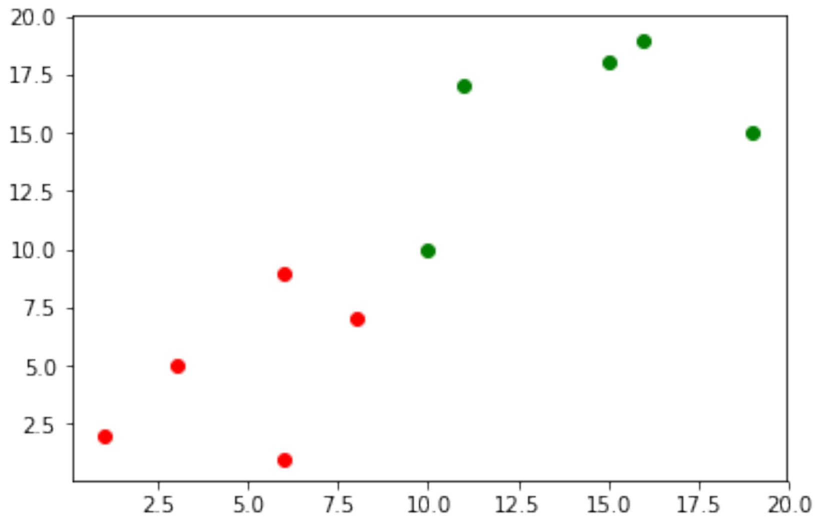
```
In [4]: X_train = np.concatenate([X1, X2])
        X_train
```

```
Out[4]: array([[ 6,  9],
               [ 6,  1],
               [ 1,  2],
               [ 8,  7],
               [ 3,  5],
               [15, 18],
               [19, 15],
               [10, 10],
               [11, 17],
               [16, 19]])
```

```
In [13]: y_train =np.array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])
         y_train
```

```
Out[13]: array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])
```

In [16]:
```
plt.scatter(X_train[y_train==0, 0], X_train[y_train==0, 1], color='
r')
plt.scatter(X_train[y_train==1, 0], X_train[y_train==1, 1], color='
g')
plt.show()
```

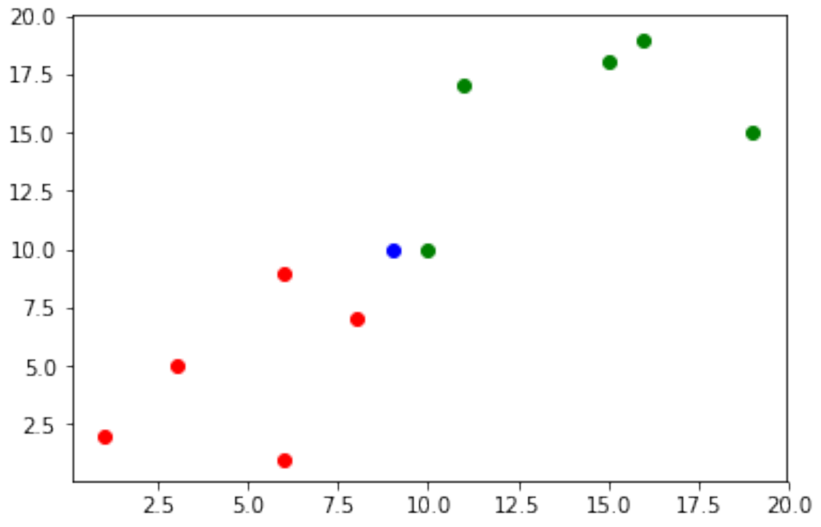


In [17]:
```
x = np.array([9, 10])
```

In [18]:
```
plt.scatter(X_train[y_train==0, 0], X_train[y_train==0, 1], color='
r')
plt.scatter(X_train[y_train==1, 0], X_train[y_train==1, 1], color='
g')

plt.scatter(x[0], x[1], color='b')

plt.show()
```



In [19]:
```
def distance(a, b, p=2):
    return np.sum(np.abs(a - b) ** p) ** (1/p)
```

```
In [21]: distances = [distance(x, item) for item in X_train]
         distances

Out[21]: [3.1622776601683795,
          9.486832980505138,
          11.313708498984761,
          3.1622776601683795,
          7.810249675906654,
          10.0,
          11.180339887498949,
          1.0,
          7.280109889280518,
          11.40175425099138]

In [23]: ind = np.argsort(distances)
         ind

Out[23]: array([7, 0, 3, 8, 4, 1, 5, 6, 2, 9])

In [24]: X_train[ind]

Out[24]: array([[10, 10],
                [ 6,  9],
                [ 8,  7],
                [11, 17],
                [ 3,  5],
                [ 6,  1],
                [15, 18],
                [19, 15],
                [ 1,  2],
                [16, 19]])

In [25]: k = 3

In [28]: y_train[ind][:k]

Out[28]: array([1, 0, 0])

In [30]: from collections import Counter
         votes = Counter(y_train[ind][:k])
         votes

Out[30]: Counter({1: 1, 0: 2})

In [31]: votes.most_common(1)     # 1 代表最多的 1 个

Out[31]: [(0, 2), (1, 1)]

In [33]: predict_y = votes.most_common(1)[0][0]

In [34]: predict_y

Out[34]: 0
```