

Импортирую данные

```
In [2]: import pandas as pd
import numpy as np

df = pd.read_excel('fountains.xlsx')
coords = df[['x', 'y']].to_numpy()
loc = df['location'].values
```

Проверю районы

```
In [3]: print(np.unique(loc), df.shape)

['Левобережный' 'Нагорье' 'Правобережный'] (500, 3)
```

Схемы фонтанов и функция расстояний

```
In [4]: schemes = {
'S1_1f': np.array([[5.0, 3.5]]),
'S2_2f': np.array([[2.0, 2.0],
[8.0, 2.0]]),
'S3_3f': np.array([[2.0, 2.0],
[8.0, 2.0],
[5.0, 5.0]])
}

def min_dists(points, fountains):
    diff = points[:, None, :] - fountains[None, :, :]
    d2 = np.sum(diff**2, axis=2)
    return np.sqrt(np.min(d2, axis=1))

dists = {name: min_dists(coords, f) for name, f in schemes.items()}
```

1. Средняя полезность по схемам

```
In [5]: names = ['S1_1f', 'S2_2f', 'S3_3f']

avg_u = {}
for n in names:
    avg_u[n] = np.mean(10 / (2 + dists[n]))

print(avg_u)

{'S1_1f': np.float64(2.0644617329645616), 'S2_2f': np.float64(3.089928468250308), 'S3_3f': np.float64(3.5033464606501776)}
```

Макимизирует полезность схема с 3 фонтанамими

2. Голосование жителей

```
In [8]: U = np.stack([10 / (2 + dists[n]) for n in names], axis=1)
```

```

nf = np.array([1, 2, 3])

votes = []
for i in range(U.shape[0]):
    row = U[i]
    mx = row.max()
    tol = 1e-9
    idxs = np.where(np.abs(row - mx) <= tol)[0]
    best_idx = idxs[np.argmin(nf[idxs])]
    votes.append(best_idx)

votes = np.array(votes)
counts = {names[j]: int(np.sum(votes == j)) for j in range(len(names))}

print(counts)

```

```
{'S1_1f': 15, 'S2_2f': 392, 'S3_3f': 93}
```

В итоге побеждает схема с 2 фонтанами - у нее 392 голоса

3. Максимизация прибыли

```

In [10]: Ai = 20.0

revenue = {}
profit = {}
for j, n in enumerate(names):
    rev = np.sum(Ai / (2 + dists[n]))
    revenue[n] = rev
    profit[n] = rev - 1500 * nf[j]

print("Выручка:", revenue)
print("Прибыль:", profit)

```

```

Выручка: {'S1_1f': np.float64(2064.4617329645616), 'S2_2f': np.float64(3089.92846825
0308), 'S3_3f': np.float64(3503.3464606501775)}
Прибыль: {'S1_1f': np.float64(564.4617329645616), 'S2_2f': np.float64(89.92846825030
801), 'S3_3f': np.float64(-996.6535393498225)}

```

4. 3 фонтана, максимизация

Компания выбирает схему с большей прибылью. Запишем прибыль в виде: $\text{Profit}_s(A) = B_s + A * C_s$, где B_s учитывает жителей не из Нагорья и фиксированные затраты, C_s = сумма по жителям Нагорья $1/(2 + \text{dist}_i^s)$

```

In [11]: S0 = schemes['S3_3f']
S1_new = np.array([[5.0, 5.5],
[4.5, 4.5],
[5.5, 4.5]])

dist0 = min_dists(coords, S0)
dist1 = min_dists(coords, S1_new)

```

```

mask_N = (loc == 'Нагорье')
mask_notN = ~mask_N

C0 = np.sum(1.0 / (2 + dist0[mask_N]))
C1 = np.sum(1.0 / (2 + dist1[mask_N]))

B0 = np.sum(20.0 / (2 + dist0[mask_notN])) - 3 * 1500
B1 = np.sum(20.0 / (2 + dist1[mask_notN])) - 3 * 1500

print("C0, C1 =", C0, C1)
print("B0, B1 =", B0, B1)

```

```

C0, C1 = 37.335084448714504 40.507878548737615
B0, B1 = -1743.3552283241124 -3042.215712970352

```

Теперь найдём A, при котором прибыли равны:

$$B0 + AC0 = B1 + AC1$$

$$A*(C1 - C0) = B0 - B1$$

$$A = (B0 - B1) / (C1 - C0)$$

```

In [12]: A_thr = (B0 - B1) / (C1 - C0)
print("A_thr =", A_thr)

```

```
A_thr = 409.37433810683723
```

Проверяем для A чуть меньше и чуть больше порога

```

In [13]: def profits(A):
    P0 = B0 + A * C0
    P1 = B1 + A * C1
    return P0, P1

print("A = 400:", profits(400.0))
print("A = 410:", profits(410.0))

```

```

A = 400: (np.float64(13190.67855116169), np.float64(13160.935706524695))
A = 410: (np.float64(13564.029395648835), np.float64(13566.01449201207))

```

Да, идеальное пороговое A это 409.37