In [7]:
```python
import numpy as np
import pandas as pd
```

In [8]:
```python
def f(x: float) -> float:
    return x**2 - 15 * x + 14
```

0. Расчёт точного значения минимума функции

(см. файл Задание1_Листок.pdf)

1. Пассивный метод поиска минимума

Начальные данные

In [9]:
```python
a = 0
b = 8
n_1 = 16
e = 0.1
n_2 = 17
```

а) Чётное N

In [10]:
```python
# определение точек x_i
x_arr = []
for j in range(1, n_1 // 2 + 1):
    x_1 = a + (b - a) / (n_1 // 2 + 1) * j - e / 2
    x_2 = x_1 + e
    x_arr.append(x_1)
    x_arr.append(x_2)
print(f"x_i: {x_arr}")

# вычисление f(x_i)
f_arr = np.array([f(x) for x in x_arr])
print(f"f_i: {f_arr}")

print("\nРезультат оптимизации\n")

# получение минимального значения функции, соответсвующего этому значению икса и его индекса
min_x_ind = f_arr.argmin()
min_f = f_arr[min_x_ind]
min_x = x_arr[min_x_ind]
print(f"x_min = x_{min_x_ind + 1} = {min_x}")
print(f"f_min = f(x_{min_x_ind + 1}) = {min_f}")

# определение сегмента локализации
localization_seg = ()
if (min_x_ind == n_1 - 1):
    localization_seg = (x_arr[min_x_ind - 1], b)
elif (min_x_ind == 0):
    localization_seg = (a, x_arr[min_x_ind + 1])
else:
    localization_seg = (x_arr[min_x_ind - 1], x_arr[min_x_ind + 1])
print(f"Отрезок локализации: {localization_seg}")
```

```
x_i: [0.8388888888888888, 0.9388888888888888, 1.7277777777777776, 1.8277777777777777, 2.6166666666666667, 2.716
666666666667, 3.5055555555555555, 3.6055555555555556, 4.394444444444445, 4.494444444444445, 5.283333333333333,
5.383333333333333, 6.172222222222222, 6.272222222222221, 7.061111111111111, 7.1611111111111105]
f_i: [  2.12040123   0.79817901  -8.93145062 -10.07589506 -18.40305556
 -19.36972222 -26.29441358 -27.08330247 -32.60552469 -33.2166358
 -37.33638889 -37.76972222 -40.48700617 -40.74256173 -42.05737654
 -42.13515432]

Результат оптимизации

x_min = x_16 = 7.1611111111111105
f_min = f(x_16) = -42.13515432098765
Отрезок локализации: (7.061111111111111, 8)
```

б) Чётное N

In [11]:
```python
# определение точек x_i
x_arr = []
for i in range(1, n_2 + 1):
    x = a + (b - a) / (n_2 + 1) * i
    x_arr.append(x)
print(f"x_i: {x_arr}")

# вычисление f(x_i)
f_arr = np.array([f(x) for x in x_arr])
print(f"f_i: {f_arr}")

print("\nРезультат оптимизации\n")

# получение минимального значения функции, соответсвующего этому значению икса и его индекса
min_x_ind = f_arr.argmin()
min_f = f_arr[min_x_ind]
min_x = x_arr[min_x_ind]
print(f"x_min = x_{min_x_ind + 1} = {min_x}")
print(f"f_min = f(x_{min_x_ind + 1}) = {min_f}")

# определение сегмента локализации
localization_seg = ()
if (min_x_ind == n_2 - 1):
    localization_seg = (x_arr[min_x_ind - 1], b)
elif (min_x_ind == 0):
    localization_seg = (a, x_arr[min_x_ind + 1])
else:
    localization_seg = (x_arr[min_x_ind - 1], x_arr[min_x_ind + 1])
print(f"Отрезок локализации: {localization_seg}")
```

```
x_i: [0.4444444444444444, 0.8888888888888888, 1.3333333333333333, 1.7777777777777777, 2.2222222222222223, 2.666
6666666666665, 3.1111111111111107, 3.5555555555555554, 4.0, 4.444444444444445, 4.888888888888888, 5.33333333333
3333, 5.777777777777778, 6.222222222222221, 6.666666666666666, 7.111111111111111, 7.555555555555555]
f_i: [  7.5308642    1.45679012  -4.22222222  -9.50617284 -14.39506173
 -18.88888889 -22.98765432 -26.69135802 -30.         -32.91358025
 -35.43209877 -37.55555556 -39.28395062 -40.61728395 -41.55555556
 -42.09876543 -42.24691358]

Результат оптимизации

x_min = x_17 = 7.555555555555555
f_min = f(x_17) = -42.24691358024691
Отрезок локализации: (7.111111111111111, 8)
```

2. Метод дихотомии (половинного деления)

Начальные данные

In [12]:
```python
a = 0
b = 8
n = 16
e = 0.1
```

In [13]:
```python
# расчёт x_i и отрезка локализации
curr_a = a
curr_b = b
x_arr = []
for j in range(n // 2):
    x1 = (curr_a + curr_b) / 2 - e / 2
    x2 = x1 + e
    x_arr.append(x1)
    x_arr.append(x2)
    f1 = f(x1)
    f2 = f(x2)
    if f1 <= f2:
        curr_b = x2
    else:
        curr_a = x1
print(f"x_i: {x_arr}\n")
print(f"Отрезок локализации: ({curr_a}, {curr_b})\n")

# определение точек находящихся в отрезке локализации
in_local_x_arr = [x for x in x_arr if curr_a <= x <= curr_b]
print(f"x внутри отрезка локализации: {in_local_x_arr}\n")

# расчёт f для точек находящихся в отрезке локализации
in_local_f_arr = np.array([f(x) for x in in_local_x_arr])
print(f"f(x) для x внутри отрезка локализации: {in_local_f_arr}")

print("\nРезультат оптимизации\n")

# получение минимального значения функции, соответсвующего этому значению икса и его индекса
min_x_ind = in_local_f_arr.argmin()
min_f = in_local_f_arr[min_x_ind]
min_x = in_local_x_arr[min_x_ind]
print(f"x_min = x_{min_x_ind + 1} = {min_x}")
print(f"f_min = f(x_{min_x_ind + 1}) = {min_f}")
print(f"Отрезок локализации: ({curr_a}, {curr_b})\n")
```

```
x_i: [3.95, 4.05, 5.925, 6.0249999999999995, 6.9125000000000005, 7.0125, 7.406250000000001, 7.5062500000000005,
7.653125, 7.753125, 7.5296875000000005, 7.6296875, 7.467968750000001, 7.56796875, 7.437109375000001, 7.53710937
5000001]

Отрезок локализации: (7.437109375000001, 7.56796875)

x внутри отрезка локализации: [7.5062500000000005, 7.5296875000000005, 7.467968750000001, 7.56796875, 7.4371093
75000001, 7.537109375000001]

f(x) для x внутри отрезка локализации: [-42.24996094 -42.24911865 -42.248974   -42.24538025 -42.24604477
 -42.24862289]

Результат оптимизации

x_min = x_1 = 7.5062500000000005
f_min = f(x_1) = -42.249960937500006
Отрезок локализации: (7.437109375000001, 7.56796875)
```

3. Метод Фибоначчи

Начальные данные

In [31]:
```python
a = 0
b = 8
n = 16
e = 0.003 # при e = 0.2 не выполняется ограничение и получается неверный отрезок локализации
```

In [30]:
```python
# получение первых n + 2 числа Фибоначчи
fib_nums = [1, 1]
for i in range(2, n + 2):
    fib_nums.append(fib_nums[i-1] + fib_nums[i-2])
print(f"Числа Фибоначчи: {fib_nums}")

# проверяем что выбранное e соответствует ограничению
if e >= ((b - a) / fib_nums[-1]):
    print("\ne не соответсвует ограничению!".upper())
    print(f"{e} >= {(b - a) / fib_nums[-1]}")

# первая итерация, на которой вычисляется и x1, и x2
j = 1
curr_a = a
curr_b = b
x1 = curr_a + fib_nums[n-j-1] / fib_nums[n-j+1] * (curr_b - curr_a) - (-1) ** (n - j + 1) / fib_nums[n-j+1] * e
x2 = curr_a + fib_nums[n-j] / fib_nums[n-j+1] * (curr_b - curr_a) + (-1) ** (n - j + 1) / fib_nums[n-j+1] * e
is_left_smaller = f(x1) <= f(x2)
if is_left_smaller:
    curr_b = x2
    x2 = x1
else:
    curr_a = x1
    x1 = x2

# алгоритм Фибоначчи
while j != n - 1:
    j += 1
    if is_left_smaller:
        x1 = curr_a + fib_nums[n-j-1] / fib_nums[n-j+1] * (curr_b - curr_a) - ((-1) ** (n - j + 1)) / fib_nums[
    else:
        x2 = curr_a + fib_nums[n-j] / fib_nums[n-j+1] * (curr_b - curr_a) + ((-1) ** (n - j + 1)) / fib_nums[n-
    is_left_smaller = f(x1) <= f(x2)
    if is_left_smaller:
        curr_b = x2
        x2 = x1
    else:
        curr_a = x1
        x1 = x2

print("\nРезультат оптимизации\n")

# получение минимального значения функции, соответсвующего этому значению икса и его индекса
print(f"x_min = {x2}")
print(f"f_min = {f(x2)}")
print(f"Отрезок локализации: ({curr_a}, {curr_b})")
```

```
Числа Фибоначчи: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584]

Результат оптимизации

x_min = 7.501470882905448
f_min = -42.249997836503475
Отрезок локализации: (7.498470882905448, 7.504626174076393)
```

4. Метод золотого сечения

Начальные данные

In [32]:
```python
a = 0
b = 8
n = 16
```

In [33]:
```python
fib1 = (3 - 5 ** (1 / 2)) / 2
fib2 = (5 ** (1 / 2) - 1) / 2
print(f"Ф1 = {fib1}")
print(f"Ф2 = {fib2}")

# первая итерация, на которой вычисляется и x1, и x2
j = 1
curr_a = a
curr_b = b
x1 = curr_a + fib1 * (curr_b - curr_a)
x2 = curr_a + fib2 * (curr_b - curr_a)
is_left_smaller = f(x1) <= f(x2)
if is_left_smaller:
    curr_b = x2
    x2 = x1
else:
    curr_a = x1
    x1 = x2

# алгоритм золотого сечения
while j != n - 1:
    j += 1
    if is_left_smaller:
        x1 = curr_a + fib1 * (curr_b - curr_a)
    else:
        x2 = curr_a + fib2 * (curr_b - curr_a)
    is_left_smaller = f(x1) <= f(x2)
    if is_left_smaller:
        curr_b = x2
        x2 = x1
    else:
        curr_a = x1
        x1 = x2

print("\nРезультат оптимизации\n")

# получение минимального значения функции, соответсвующего этому значению икса и его индекса
print(f"x_min = {x2}")
print(f"f_min = {f(x2)}")
print(f"Отрезок локализации: ({curr_a}, {curr_b})")
```

```
Ф1 = 0.3819660112501051
Ф2 = 0.6180339887498949

Результат оптимизации

x_min = 7.500860528920722
f_min = -42.249999259489975
Отрезок локализации: (7.498620260264141, 7.504485359751)
```