LAMPIRAN

LAMPIRAN A- 1 *Source code* metode *Modified Revised Ones Assignment* (MROA) dengan *software python programming.*

```
import numpy as np
from scipy.optimize import linear_sum_assignment

# ---------- Utilitas tampil ----------
def print_vec(title, vec):
   print(title, "[" + ", ".join(f"{v:.0f}" if float(v).is_integer() else f"{v:.3f}" for v
in np.asarray(vec)) + "]")

def print_mat(title, M):
   print(title)
   for i in range(M.shape[0]):
      row = " ".join(f"{x:.0f}" if float(x).is_integer() else f"{x:.3f}" for x in
M[i])
      print(f"  {row}")

# ---------- Proses dengan LOOP ----------
def find_max_with_loops(M):
   max_val = M[0, 0]
   print("\n[Loop] Cari max matrix (scan elemen satu per satu):")
   for i in range(M.shape[0]):
      for j in range(M.shape[1]):
         if M[i, j] > max_val:
            max_val = M[i, j]
            print(f"  -> update max: M[{i+1},{j+1}] = {M[i,j]}")
   print(f"Hasil max_val = {max_val}")
   return max_val

def build_cost_for_max_with_loops(M, max_val):
   C = np.zeros_like(M, dtype=float)
   print("\n[Loop] Bangun cost_matrix_max = max_val - matrix:")
   for i in range(M.shape[0]):
      for j in range(M.shape[1]):
         C[i, j] = max_val - M[i, j]
   print_mat("cost_matrix_max:", C)
   return C

def row_reduction_with_loops(C):
   R = C.copy().astype(float)
   mins = np.zeros(R.shape[0], dtype=float)
   print("\n[Loop] Row reduction (kurangi tiap baris dg nilai minimum baris):")
   for i in range(R.shape[0]):
      # cari min baris manual
      m = R[i, 0]
      for j in range(1, R.shape[1]):
         if R[i, j] < m:
```

```
            m = R[i, j]
        mins[i] = m
        before = R[i, :].copy()
        # kurangi
        for j in range(R.shape[1]):
            R[i, j] = R[i, j] - m
        print(f"  Baris {i+1}: min={m}")
        print("    sebelum:", " ".join(f"{x:.0f}" if float(x).is_integer() else
f"{x:.3f}" for x in before))
        print("    sesudah:", " ".join(f"{x:.0f}" if float(x).is_integer() else
f"{x:.3f}" for x in R[i, :]))
    print_vec("Min per baris:", mins)
    return R, mins

def col_reduction_with_loops(R):
    C = R.copy().astype(float)
    mins = np.zeros(C.shape[1], dtype=float)
    print("\n[Loop] Column reduction (kurangi tiap kolom dg nilai minimum
kolom):")
    for j in range(C.shape[1]):
        # cari min kolom manual
        m = C[0, j]
        for i in range(1, C.shape[0]):
            if C[i, j] < m:
                m = C[i, j]
        mins[j] = m
        before = C[:, j].copy()
        # kurangi
        for i in range(C.shape[0]):
            C[i, j] = C[i, j] - m
        print(f"  Kolom {j+1}: min={m}")
        print("    sebelum:", " ".join(f"{x:.0f}" if float(x).is_integer() else
f"{x:.3f}" for x in before))
        print("    sesudah:", " ".join(f"{x:.0f}" if float(x).is_integer() else
f"{x:.3f}" for x in C[:, j]))
    print_vec("Min per kolom:", mins)
    return C, mins

def count_zeros_with_loops(A):
    zr = np.zeros(A.shape[0], dtype=int)
    zc = np.zeros(A.shape[1], dtype=int)
    pos = []
    print("\n[Loop] Hitung nol per baris & kolom + posisi nol:")
    for i in range(A.shape[0]):
        for j in range(A.shape[1]):
            if abs(A[i, j]) < 1e-12:
                zr[i] += 1
                zc[j] += 1
```

```
                pos.append((i+1, j+1))
    print_vec("Jumlah nol per baris:", zr)
    print_vec("Jumlah nol per kolom:", zc)
    print("Posisi nol (baris, kolom):", pos)
    return zr, zc, pos

# ---------- Solver + tampilan proses ----------
def mroa_assignment_with_loops(matrix):
    matrix = np.asarray(matrix)
    print("=== INFORMASI AWAL ===")
    print(f"Bentuk matriks: {matrix.shape}, dtype: {matrix.dtype}")
    print_mat("Matriks input:", matrix)

    # ========== PROSES & HASIL: MAKSIMUM ==========
    print("\n================ MAKSIMUM ================")
    max_val = find_max_with_loops(matrix)
    cost_max = build_cost_for_max_with_loops(matrix, max_val)
    rred_max, rmins_max = row_reduction_with_loops(cost_max)
    cred_max, cmins_max = col_reduction_with_loops(rred_max)
    count_zeros_with_loops(cred_max)

    print("\n[Solver] Jalankan cost_matrix_max (tanpa ubah algoritma)")
    row_ind_max, col_ind_max = linear_sum_assignment(cost_max)
    assignment_matrix_max = np.zeros_like(matrix, dtype=int)
    total_value_max = 0
    print("\n=== HASIL Penugasan Maksimum (loop per pasangan) ===")
    for step, (w, t) in enumerate(zip(row_ind_max, col_ind_max), start=1):
        assignment_matrix_max[w, t] = 1
        val = matrix[w, t]
        total_value_max += val
        print(f"  Step-{step}: Pekerja {w+1} → Tugas {t+1} (Nilai: {val})")
    print(f"Total nilai maksimum: {total_value_max}")
    print_mat("Matriks penugasan maksimum (1=ditugaskan):",
assignment_matrix_max)

    # ========== PROSES & HASIL: MINIMUM ==========
    print("\n================ MINIMUM ================")
    cost_min = matrix.astype(float)
    print_mat("[Loop] cost_matrix_min = matrix (biaya asli):", cost_min)
    rred_min, rmins_min = row_reduction_with_loops(cost_min)
    cred_min, cmins_min = col_reduction_with_loops(rred_min)
    count_zeros_with_loops(cred_min)

    print("\n[Solver] Jalankan cost_matrix_min (tanpa ubah algoritma)")
    row_ind_min, col_ind_min = linear_sum_assignment(cost_min)
    assignment_matrix_min = np.zeros_like(matrix, dtype=int)
    total_cost_min = 0
    print("\n=== HASIL Penugasan Minimum (loop per pasangan) ===")
```

```
    for step, (w, t) in enumerate(zip(row_ind_min, col_ind_min), start=1):
        assignment_matrix_min[w, t] = 1
        cost = matrix[w, t]
        total_cost_min += cost
        print(f"  Step-{step}: Pekerja {w+1} → Tugas {t+1} (Biaya: {cost})")
    print(f"Total biaya minimum: {total_cost_min}")
    print_mat("Matriks penugasan minimum (1=ditugaskan):",
assignment_matrix_min)
```

LAMPIRAN A- 2 Hasil hasil uji optimalisasi dengan matriks 15x15 berbantu *Python Programming*.

```
>>> Penugasan untuk Matriks 15x15:
=== INFORMASI AWAL ===
Bentuk matriks: (15, 15), dtype: int64 Matriks
input:
  12 8 11 18 11 15 17 19 14 10 16 13 21 20 9
  14 22 8 12 14 18 20 15 19 11 10 17 22 21 13
  14 14 16 14 15 12 13 11 18 19 20 22 21 16 10
  19 11 14 17 15 18 20 13 14 12 15 11 10 22 19
  13 9 17 20 11 19 21 14 18 15 22 16 13 11 12
  11 15 18 20 19 13 10 22 14 17 21 12 20 18 16
  18 20 19 13 10 16 15 11 22 21 17 14 19 13 12
  15 17 20 18 16 21 13 10 12 22 19 15 11 14 20
  13 10 22 21 17 19 14 18 15 16 11 12 20 13 19
  20 19 13 10 12 16 18 15 22 21 14 17 11 20 13
  19 13 10 12 16 18 15 22 21 14 17 11 20 13 19
  22 21 14 17 11 20 13 19 10 12 16 18 15 22 21
  16 18 15 22 21 14 17 11 20 13 19 10 12 16 18
  15 22 21 14 17 11 20 13 19 10 12 16 18 15 22
  14 17 11 20 13 19 10 12 16 18 15 22 21 14 17

================ MAKSIMUM ================

[Loop] Cari max matrix (scan elemen satu per satu):
  -> update max: M[1,4] = 18
  -> update max: M[1,8] = 19
  -> update max: M[1,13] = 21
  -> update max: M[2,2] = 22 Hasil
max_val = 22

[Loop] Bangun cost_matrix_max = max_val - matrix: cost_matrix_max:
  10 14 11 4 11 7 5 3 8 12 6 9 1 2 13
  8 0 14 10 8 4 2 7 3 11 12 5 0 1 9
  8 8 6 8 7 10 9 11 4 3 2 0 1 6 12
```

3 11 8 5 7 4 2 9 8 10 7 11 12 0 3
9 13 5 2 11 3 1 8 4 7 0 6 9 11 10
11 7 4 2 3 9 12 0 8 5 1 10 2 4 6
4 2 3 9 12 6 7 11 0 1 5 8 3 9 10
7 5 2 4 6 1 9 12 10 0 3 7 11 8 2
9 12 0 1 5 3 8 4 7 6 11 10 2 9 3
2 3 9 12 10 6 4 7 0 1 8 5 11 2 9
3 9 12 10 6 4 7 0 1 8 5 11 2 9 3
0 1 8 5 11 2 9 3 12 10 6 4 7 0 1
6 4 7 0 1 8 5 11 2 9 3 12 10 6 4
7 0 1 8 5 11 2 9 3 12 10 6 4 7 0
8 5 11 2 9 3 12 10 6 4 7 0 1 8 5

[Loop] Row reduction (kurangi tiap baris dg nilai minimum baris):
Baris 1: min=1.0
sebelum: 10 14 11 4 11 7 5 3 8 12 6 9 1 2 13
sesudah: 9 13 10 3 10 6 4 2 7 11 5 8 0 1 12
Baris 2: min=0.0
sebelum: 8 0 14 10 8 4 2 7 3 11 12 5 0 1 9
sesudah: 8 0 14 10 8 4 2 7 3 11 12 5 0 1 9
Baris 3: min=0.0
sebelum: 8 8 6 8 7 10 9 11 4 3 2 0 1 6 12
sesudah: 8 8 6 8 7 10 9 11 4 3 2 0 1 6 12
Baris 4: min=0.0
sebelum: 3 11 8 5 7 4 2 9 8 10 7 11 12 0 3
sesudah: 3 11 8 5 7 4 2 9 8 10 7 11 12 0 3
Baris 5: min=0.0
sebelum: 9 13 5 2 11 3 1 8 4 7 0 6 9 11 10
sesudah: 9 13 5 2 11 3 1 8 4 7 0 6 9 11 10
Baris 6: min=0.0
sebelum: 11 7 4 2 3 9 12 0 8 5 1 10 2 4 6
sesudah: 11 7 4 2 3 9 12 0 8 5 1 10 2 4 6
Baris 7: min=0.0
sebelum: 4 2 3 9 12 6 7 11 0 1 5 8 3 9 10
sesudah: 4 2 3 9 12 6 7 11 0 1 5 8 3 9 10
Baris 8: min=0.0
sebelum: 7 5 2 4 6 1 9 12 10 0 3 7 11 8 2
sesudah: 7 5 2 4 6 1 9 12 10 0 3 7 11 8 2
Baris 9: min=0.0
sebelum: 9 12 0 1 5 3 8 4 7 6 11 10 2 9 3
sesudah: 9 12 0 1 5 3 8 4 7 6 11 10 2 9 3
Baris 10: min=0.0
sebelum: 2 3 9 12 10 6 4 7 0 1 8 5 11 2 9
sesudah: 2 3 9 12 10 6 4 7 0 1 8 5 11 2 9
Baris 11: min=0.0
sebelum: 3 9 12 10 6 4 7 0 1 8 5 11 2 9 3

```
    sesudah: 3 9 12 10 6 4 7 0 1 8 5 11 2 9 3
  Baris 12: min=0.0
    sebelum: 0 1 8 5 11 2 9 3 12 10 6 4 7 0 1
    sesudah: 0 1 8 5 11 2 9 3 12 10 6 4 7 0 1
  Baris 13: min=0.0
    sebelum: 6 4 7 0 1 8 5 11 2 9 3 12 10 6 4
    sesudah: 6 4 7 0 1 8 5 11 2 9 3 12 10 6 4
  Baris 14: min=0.0
    sebelum: 7 0 1 8 5 11 2 9 3 12 10 6 4 7 0
    sesudah: 7 0 1 8 5 11 2 9 3 12 10 6 4 7 0
  Baris 15: min=0.0
    sebelum: 8 5 11 2 9 3 12 10 6 4 7 0 1 8 5
    sesudah: 8 5 11 2 9 3 12 10 6 4 7 0 1 8 5
Min per baris: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[Loop] Column reduction (kurangi tiap kolom dg nilai minimum kolom):
  Kolom 1: min=0.0
    sebelum: 9 8 8 3 9 11 4 7 9 2 3 0 6 7 8
    sesudah: 9 8 8 3 9 11 4 7 9 2 3 0 6 7 8
  Kolom 2: min=0.0
    sebelum: 13 0 8 11 13 7 2 5 12 3 9 1 4 0 5
    sesudah: 13 0 8 11 13 7 2 5 12 3 9 1 4 0 5
  Kolom 3: min=0.0
    sebelum: 10 14 6 8 5 4 3 2 0 9 12 8 7 1 11
    sesudah: 10 14 6 8 5 4 3 2 0 9 12 8 7 1 11
  Kolom 4: min=0.0
    sebelum: 3 10 8 5 2 2 9 4 1 12 10 5 0 8 2
    sesudah: 3 10 8 5 2 2 9 4 1 12 10 5 0 8 2
  Kolom 5: min=1.0
    sebelum: 10 8 7 7 11 3 12 6 5 10 6 11 1 5 9
    sesudah: 9 7 6 6 10 2 11 5 4 9 5 10 0 4 8
  Kolom 6: min=1.0
    sebelum: 6 4 10 4 3 9 6 1 3 6 4 2 8 11 3
    sesudah: 5 3 9 3 2 8 5 0 2 5 3 1 7 10 2
  Kolom 7: min=1.0
    sebelum: 4 2 9 2 1 12 7 9 8 4 7 9 5 2 12
    sesudah: 3 1 8 1 0 11 6 8 7 3 6 8 4 1 11
  Kolom 8: min=0.0
    sebelum: 2 7 11 9 8 0 11 12 4 7 0 3 11 9 10
    sesudah: 2 7 11 9 8 0 11 12 4 7 0 3 11 9 10
  Kolom 9: min=0.0
    sebelum: 7 3 4 8 4 8 0 10 7 0 1 12 2 3 6
    sesudah: 7 3 4 8 4 8 0 10 7 0 1 12 2 3 6
  Kolom 10: min=0.0
    sebelum: 11 11 3 10 7 5 1 0 6 1 8 10 9 12 4
    sesudah: 11 11 3 10 7 5 1 0 6 1 8 10 9 12 4
```

Kolom 11: min=0.0
sebelum: 5 12 2 7 0 1 5 3 11 8 5 6 3 10 7
sesudah: 5 12 2 7 0 1 5 3 11 8 5 6 3 10 7
Kolom 12: min=0.0
sebelum: 8 5 0 11 6 10 8 7 10 5 11 4 12 6 0
sesudah: 8 5 0 11 6 10 8 7 10 5 11 4 12 6 0
Kolom 13: min=0.0
sebelum: 0 0 1 12 9 2 3 11 2 11 2 7 10 4 1
sesudah: 0 0 1 12 9 2 3 11 2 11 2 7 10 4 1
Kolom 14: min=0.0
sebelum: 1 1 6 0 11 4 9 8 9 2 9 0 6 7 8
sesudah: 1 1 6 0 11 4 9 8 9 2 9 0 6 7 8
Kolom 15: min=0.0
sebelum: 12 9 12 3 10 6 10 2 3 9 3 1 4 0 5
sesudah: 12 9 12 3 10 6 10 2 3 9 3 1 4 0 5
Min per kolom: [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]

[Loop] Hitung nol per baris & kolom + posisi nol:
Jumlah nol per baris: [1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 1, 2, 2, 2, 1]
Jumlah nol per kolom: [1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1]
Posisi nol (baris, kolom): [(1, 13), (2, 2), (2, 13), (3, 12), (4, 14), (5, 7), (5, 11), (6, 8), (7, 9), (8, 6), (8, 10), (9, 3), (10, 9), (11, 8), (12, 1), (12, 14), (13, 4), (13, 5), (14, 2), (14, 15), (15, 12)]

[Solver] Jalankan cost_matrix_max (tanpa ubah algoritma)

=== HASIL Penugasan Maksimum (loop per pasangan) ===
Step-1: Pekerja 1 → Tugas 13 (Nilai: 21)
Step-2: Pekerja 2 → Tugas 2 (Nilai: 22)
Step-3: Pekerja 3 → Tugas 12 (Nilai: 22)
Step-4: Pekerja 4 → Tugas 14 (Nilai: 22)
Step-5: Pekerja 5 → Tugas 7 (Nilai: 21)
Step-6: Pekerja 6 → Tugas 11 (Nilai: 21)
Step-7: Pekerja 7 → Tugas 9 (Nilai: 22)
Step-8: Pekerja 8 → Tugas 6 (Nilai: 21)
Step-9: Pekerja 9 → Tugas 3 (Nilai: 22)
Step-10: Pekerja 10 → Tugas 10 (Nilai: 21)
Step-11: Pekerja 11 → Tugas 8 (Nilai: 22)
Step-12: Pekerja 12 → Tugas 1 (Nilai: 22)
Step-13: Pekerja 13 → Tugas 5 (Nilai: 21)
Step-14: Pekerja 14 → Tugas 15 (Nilai: 22)
Step-15: Pekerja 15 → Tugas 4 (Nilai: 20)
Total nilai maksimum: 322

Matriks penugasan maksimum
(1=ditugaskan): 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

LAMPIRAN A- 3 Hasil hasil uji optimalisasi dengan matriks 5x5 berbantu Python Programming.

```
>>> Penugasan untuk Matriks 5x5:
=== INFORMASI AWAL ===
Bentuk matriks: (5, 5), dtype: int64
Matriks input:
  7 8 4 15 12
  7 9 1 14 10
  9 1 1 6 7
  7 6 14 6 10
  1 6 12 10 6

================ MAKSIMUM ================

[Loop] Cari max matrix (scan elemen satu per satu):
  -> update max: M[1,2] = 8
  -> update max: M[1,4] = 15
Hasil max_val = 15

[Loop] Bangun cost_matrix_max = max_val - matrix:
cost_matrix_max:
  8 7 11 0 3
  8 6 14 1 5
  6 14 14 9 8
  8 9 1 9 5
  14 9 3 5 9

[Loop] Row reduction (kurangi tiap baris dg nilai minimum baris):
  Baris 1: min=0.0
    sebelum: 8 7 11 0 3
    sesudah: 8 7 11 0 3
  Baris 2: min=1.0
    sebelum: 8 6 14 1 5
    sesudah: 7 5 13 0 4
  Baris 3: min=6.0
    sebelum: 6 14 14 9 8
    sesudah: 0 8 8 3 2
  Baris 4: min=1.0
    sebelum: 8 9 1 9 5
    sesudah: 7 8 0 8 4
  Baris 5: min=3.0
    sebelum: 14 9 3 5 9
    sesudah: 11 6 0 2 6
Min per baris: [0, 1, 6, 1, 3]

[Loop] Column reduction (kurangi tiap kolom dg nilai minimum kolom):
  Kolom 1: min=0.0
    sebelum: 8 7 0 7 11
```

```
    sesudah: 8 7 0 7 11
  Kolom 2: min=5.0
    sebelum: 7 5 8 8 6
    sesudah: 2 0 3 3 1
  Kolom 3: min=0.0
    sebelum: 11 13 8 0 0
    sesudah: 11 13 8 0 0
  Kolom 4: min=0.0
    sebelum: 0 0 3 8 2
    sesudah: 0 0 3 8 2
  Kolom 5: min=2.0
    sebelum: 3 4 2 4 6
    sesudah: 1 2 0 2 4
Min per kolom: [0, 5, 0, 0, 2]

[Loop] Hitung nol per baris & kolom + posisi nol:
Jumlah nol per baris: [1, 2, 2, 1, 1]
Jumlah nol per kolom: [1, 1, 2, 2, 1]
Posisi nol (baris, kolom): [(1, 4), (2, 2), (2, 4), (3, 1), (3, 5), (4, 3), (5, 3)]

[Solver] Jalankan cost_matrix_max (tanpa ubah algoritma)

=== HASIL Penugasan Maksimum (loop per pasangan) ===
  Step-1: Pekerja 1 → Tugas 5 (Nilai: 12)
  Step-2: Pekerja 2 → Tugas 4 (Nilai: 14)
  Step-3: Pekerja 3 → Tugas 1 (Nilai: 9)
  Step-4: Pekerja 4 → Tugas 3 (Nilai: 14)
  Step-5: Pekerja 5 → Tugas 2 (Nilai: 6)
Total nilai maksimum: 55
Matriks penugasan maksimum (1=ditugaskan):
  0 0 0 0 1
  0 0 0 1 0
  1 0 0 0 0
  0 0 1 0 0
  0 1 0 0 0
```