

INSTRUKSI TUGAS BESAR

Mata Kuliah: Riset Operasi

A. PEMBENTUKAN KELOMPOK

1. **Jumlah Anggota:** Setiap kelompok terdiri dari **3 (tiga) mahasiswa**
 2. **Pemilihan Anggota:** Mahasiswa bebas memilih anggota kelompok sendiri
 3. **Pendaftaran:**
 - Daftarkan nama kelompok dan anggota ke dosen pengampu
 - Format: Kelompok [Nomor] - [Nama Anggota 1], [Nama Anggota 2], [Nama Anggota 3]
 - Batas waktu pendaftaran: **[Tanggal yang ditentukan]**
-

B. PEMILIHAN ALGORITMA

Kategori Algoritma yang Tersedia:

1. Pemrograman Linear

- Metode Grafik
- Metode Simpleks
- **Metode Big M**
- **Metode Dua Fase (Two-Phase Method)**
- Algoritma Dual Simpleks

2. Pemrograman Bilangan Bulat

- **Algoritma Branch and Bound**
- **Algoritma Cutting Plane (Gomory Cut)**
- Metode Branch and Cut

3. Pemrograman Tak Linear

- Metode Gradient Descent
- Metode Newton-Raphson
- **Algoritma Lagrange Multiplier**
- **Metode KKT (Karush-Kuhn-Tucker)**

4. Pemrograman Kuadratik

- Metode Wolfe
- Algoritma Active Set
- Interior Point Methods

5. Pemrograman Dinamika Deterministik

- Forward Recursion
- Backward Recursion
- Bellman Equation

6. Sistem Antrian

- Model M/M/1
- Model M/M/c
- Simulasi Monte Carlo untuk Antrian

7. Analisa Jaringan

- Algoritma Dijkstra (Shortest Path)
- Algoritma Floyd-Warshall (All-Pairs Shortest Path)
- Algoritma Bellman-Ford
- Algoritma Prim (Minimum Spanning Tree)
- Algoritma Kruskal (Minimum Spanning Tree)
- Ford-Fulkerson / Edmonds-Karp (Maximum Flow)
- CPM (Critical Path Method)
- PERT (Program Evaluation Review Technique)

8. Line Balancing

- Ranked Positional Weight Method
- Longest Operation Time (LOT)
- Kilbridge-Wester Heuristic

9. Teori Keputusan

- Algoritma Minimax/Maximin
- Expected Value Criterion
- Decision Tree Analysis
- Bayesian Decision Theory

CATATAN : YANG MERAH YANG WAJIB DI PILIH

C. KETENTUAN PEMILIHAN

1. **Setiap kelompok memilih 1 (satu) algoritma** dari daftar di atas
 2. **Tidak boleh ada algoritma yang sama** antar kelompok (first come, first served)
 3. **Konfirmasi pilihan** algoritma kepada dosen setelah pembentukan kelompok
 4. Jika algoritma yang diinginkan sudah dipilih kelompok lain, pilih algoritma alternatif
-

D. TEORI DASAR YANG HARUS DIPAHAMI

Sebelum mengimplementasikan algoritma, mahasiswa **WAJIB** memahami teori dasar yang membentuk algoritma tersebut:

PANDUAN BELAJAR TEORI DASAR

Agar pembelajaran tidak melebar, ikuti instruksi ini:

1. FOKUS PADA TEORI YANG RELEVAN SAJA

- Pelajari HANYA teori yang tercantum di bawah untuk algoritma Anda
- Jangan mempelajari teori untuk algoritma lain
- Batasi referensi maksimal 3-5 sumber utama

2. GUNAKAN PETA BELAJAR (LEARNING PATH)

- Mulai dari konsep paling dasar → konsep lanjutan
- Pastikan memahami konsep sebelumnya sebelum lanjut
- Gunakan checklist untuk tracking progress

3. METODE PEMBELAJARAN BERTAHAP

- **Tahap 1 (Hari 1-2):** Baca dan pahami definisi & konsep dasar
- **Tahap 2 (Hari 3-4):** Lihat contoh dan visualisasi
- **Tahap 3 (Hari 5-6):** Kerjakan latihan manual (hitung tangan)
- **Tahap 4 (Hari 7):** Buat rangkuman & mind map

4. BATASAN KEDALAMAN MATERI

- Pahami **WHAT** (apa itu), **WHY** (kenapa digunakan), **HOW** (bagaimana cara kerja)
- Tidak perlu membuktikan teorema matematika secara formal
- Fokus pada pemahaman aplikatif, bukan teoritis murni

5. SUMBER BELAJAR YANG DIREKOMENDASIKAN

- **Buku:** "Introduction to Operations Research" - Hillier & Lieberman

- **Buku:** "Operations Research: An Introduction" - Taha
- **Online:** GeeksforGeeks, YouTube (Abdul Bari, MIT OCW)
- **Video:** 3Blue1Brown untuk visualisasi matematika
- **Bahasa:** Boleh campuran Indonesia & Inggris

6. INDIKATOR PEMAHAMAN CUKUP

- Bisa menjelaskan dengan kata-kata sendiri
- Bisa memberikan contoh sederhana
- Bisa menghitung/simulasi manual untuk kasus kecil
- Bisa menjawab quiz self-assessment minimal 7/10 benar

7. JIKA KESULITAN

- Diskusikan dengan anggota kelompok lain
- Konsultasi dengan dosen di jam konsultasi
- Cari video tutorial berbahasa Indonesia
- Jangan terjebak di satu konsep > 2 hari

1. PEMROGRAMAN LINEAR

📍 **LEARNING PATH:** Aljabar Linear → Sistem Persamaan → Geometri → Metode Penyelesaian

Teori Dasar yang Harus Dipelajari:

Aljabar Linear - [2 hari]

- **Vektor dan Matriks**
 - Operasi vektor (penjumlahan, perkalian skalar, dot product)
 - Operasi matriks (penjumlahan, perkalian, transpose, invers)
 - **BUATKAN:** Algoritma operasi matriks dasar
 - **PSEUDOCODE:** Perkalian dua matriks
- **Sistem Persamaan Linear**
 - Metode eliminasi Gauss
 - Metode Gauss-Jordan
 - Representasi matriks augmented
 - **BUATKAN:** Algoritma eliminasi Gauss
 - **PSEUDOCODE:** Gauss-Jordan untuk mencari solusi

- **✗ SKIP:** Eigenvalues, eigenvectors, determinan (tidak relevan untuk LP dasar)

Geometri Komputasi Dasar - [1 hari]

- **✓ Representasi Geometri**
 - Koordinat kartesian 2D dan 3D
 - Persamaan garis dan hyperplane
 - Feasible region dan extreme points
 - **BUATKAN:** Algoritma mencari titik potong dua garis
 - **PSEUDOCODE:** Identifikasi feasible region
- **✗ SKIP:** Convex hull algorithms (Graham scan, Jarvis march)

Teori Optimasi Dasar - [2 hari]

- **✓ Fungsi Objektif dan Konstrain**
 - Fungsi linear: $f(x) = c_1x_1 + c_2x_2 + \dots + c_nx_n$
 - Inequality constraints (\leq, \geq)
 - Equality constraints (=)
 - **BUATKAN:** Algoritma evaluasi fungsi objektif
 - **PSEUDOCODE:** Checking constraint feasibility
- **✓ Kondisi Optimalitas**
 - Basic feasible solution
 - Optimal solution criteria
 - Degeneracy dan unbounded solutions
- **✗ SKIP:** KKT conditions (untuk nonlinear), Lagrangian (untuk constrained optimization)

Metode Simpleks - [3 hari]

- **✓ Bentuk Standar LP**
 - Konversi min/max
 - Slack, surplus, artificial variables
 - **BUATKAN:** Algoritma konversi ke bentuk standar
 - **PSEUDOCODE:** Adding slack/surplus variables
- **✓ Tableau Simpleks**
 - Initial tableau construction
 - Pivot operation

- Optimality test
- **BUATKAN:** Algoritma pivot operation
- **PSEUDOCODE:** Simplex iteration
- **✗ SKIP:** Revised simplex, dual simplex (kecuali dipilih)

Teori Dualitas (untuk Dual Simplex) - [2 hari]

- **✓ Primal-Dual Relationship**
 - Constructing dual problem
 - Weak duality theorem
 - Strong duality theorem (konsep saja)
 - Complementary slackness
 - **BUATKAN:** Algoritma konstruksi dual problem
 - **PSEUDOCODE:** Primal to dual conversion
- **✗ SKIP:** Shadow prices interpretation

Latihan Wajib:

- Selesaikan 3 masalah LP dengan metode grafik (2 variabel)
- Hitung manual 2 iterasi metode simpleks
- Konversi 1 primal problem ke dual

Algoritma: Metode Grafik, Metode Simpleks, Big M, Two-Phase, Dual Simplex

2. PEMROGRAMAN BILANGAN BULAT

 **LEARNING PATH:** LP Relaxation → Tree Structure → Branch and Bound → Cutting Plane

Teori Dasar yang Harus Dipelajari:

Teori Graf (Tree Structure) - [2 hari]

- **✓ Tree Data Structure**
 - Node, edge, parent-child relationship
 - Binary tree, n-ary tree
 - Tree traversal (DFS, BFS)
 - **BUATKAN:** Algoritma DFS untuk tree
 - **PSEUDOCODE:** BFS traversal
- **✓ Directed Acyclic Graph (DAG)**

- Path enumeration
- Level-order processing
- **✗ SKIP:** AVL trees, Red-Black trees

LP Relaxation - [1 hari]

- **✓ Relaksasi Batasan Integer**
 - Continuous relaxation
 - LP bound vs IP optimal
 - Rounding techniques
 - **BUATKAN:** Algoritma LP relaxation
 - **PSEUDOCODE:** Solve relaxed LP
- **✗ SKIP:** Lagrangian relaxation

Branch and Bound - [3 hari]

- **✓ Branching Strategy**
 - Variable selection
 - Node selection (best-first, depth-first, breadth-first)
 - **BUATKAN:** Algoritma Branch and Bound
 - **PSEUDOCODE:** Complete B&B procedure
- **✓ Bounding**
 - Upper bound dan lower bound
 - Pruning conditions
 - Fathoming rules
 - **BUATKAN:** Algoritma bounding check
 - **PSEUDOCODE:** Pruning decision
- **✗ SKIP:** Advanced branching rules (strong branching)

Cutting Plane (Gomory Cut) - [2 hari]

- **✓ Valid Inequalities**
 - Fractional Gomory cuts
 - Cut generation from tableau
 - **BUATKAN:** Algoritma Gomory cut generation
 - **PSEUDOCODE:** Add cut to LP

- **Separation Problem**
 - Identifying violated constraints
- **SKIP:** Chvatal-Gomory cuts, lift-and-project

 **Latihan Wajib:**

- Selesaikan 1 IP problem dengan B&B (manual, max 3 levels)
- Generate 2 Gomory cuts dari fractional solution
- Trace branching tree untuk masalah kecil

Algoritma: Branch and Bound, Cutting Plane (Gomory), Branch and Cut

3. PEMROGRAMAN TAK LINEAR

 **LEARNING PATH:** Kalkulus Multivariabel → Gradient → Newton Method → Constrained Optimization

Teori Dasar yang Harus Dipelajari:

Kalkulus Multivariabel - [3 hari]

- **Turunan Parsial**
 - $\partial f / \partial x, \partial f / \partial y$ untuk fungsi 2 variabel
 - Gradient vector: $\nabla f = [\partial f / \partial x_1, \partial f / \partial x_2, \dots, \partial f / \partial x_n]$
 - **BUATKAN:** Algoritma hitung gradient numerik
 - **PSEUDOCODE:** Numerical differentiation
- **Turunan Kedua**
 - Hessian matrix: $H(f)$
 - Second-order conditions
 - **BUATKAN:** Algoritma hitung Hessian (numerical)
 - **PSEUDOCODE:** Hessian construction
- **Taylor Series**
 - First-order approximation: $f(x+\Delta x) \approx f(x) + \nabla f \cdot \Delta x$
 - Second-order approximation
- **SKIP:** Higher-order derivatives, implicit differentiation

Metode Numerik - [2 hari]

- **Iterasi dan Konvergensi**
 - Fixed-point iteration

- Convergence criteria (tolerance)
- Rate of convergence
- **BUATKAN:** Algoritma convergence check
- **PSEUDOCODE:** Iterative solver template
- **Line Search**
 - Exact line search
 - Backtracking line search
 - Armijo condition
 - **BUATKAN:** Algoritma backtracking line search
 - **PSEUDOCODE:** Step size selection
- **SKIP:** Trust region methods

Gradient Descent - [2 hari]

- **Basic Gradient Descent**
 - Update rule: $x_{k+1} = x_k - \alpha \nabla f(x_k)$
 - Learning rate selection
 - **BUATKAN:** Algoritma Gradient Descent lengkap
 - **PSEUDOCODE:** GD with line search
- **Variants**
 - Batch GD vs Stochastic GD (konsep)
 - Momentum (konsep)
- **SKIP:** Adam, RMSprop, AdaGrad

Newton-Raphson Method - [2 hari]

- **Newton's Method**
 - Update rule: $x_{k+1} = x_k - [H(x_k)]^{-1} \nabla f(x_k)$
 - Quadratic convergence
 - **BUATKAN:** Algoritma Newton-Raphson
 - **PSEUDOCODE:** Newton iteration
- **SKIP:** Quasi-Newton methods (BFGS, L-BFGS)

Constrained Optimization - [3 hari]

- **Lagrange Multipliers**

- Lagrangian function: $L(x,\lambda) = f(x) + \lambda g(x)$
- First-order necessary conditions
- **BUATKAN:** Algoritma method of Lagrange multipliers
- **PSEUDOCODE:** Solve for stationary points
- **KKT Conditions**
 - Stationarity: $\nabla f + \lambda \nabla g + \mu \nabla h = 0$
 - Primal feasibility
 - Dual feasibility
 - Complementary slackness: $\mu_i h_i = 0$
 - **BUATKAN:** Algoritma KKT condition check
 - **PSEUDOCODE:** KKT solver
- **SKIP:** Penalty methods, barrier methods

Latihan Wajib:

- Hitung gradient dan Hessian untuk 2 fungsi
- Jalankan 5 iterasi Gradient Descent manual
- Selesaikan 1 constrained problem dengan Lagrange

Algoritma: Gradient Descent, Newton-Raphson, Lagrange Multiplier, KKT

4. PEMROGRAMAN KUADRATIK

LEARNING PATH: Quadratic Forms → Convexity → Active Set → Interior Point

Teori Dasar yang Harus Dipelajari:

Aljabar Linear Lanjutan - [2 hari]

- **Quadratic Forms**
 - $f(x) = \frac{1}{2}x^T Q x + c^T x$
 - Positive definite, semi-definite matrices
 - Eigenvalues untuk classification
 - **BUATKAN:** Algoritma evaluasi quadratic form
 - **PSEUDOCODE:** Check positive definiteness
- **Matrix Decomposition**
 - Cholesky decomposition (konsep)
 - LU decomposition (konsep)

- X **SKIP:** SVD, QR decomposition

Convex Analysis - [2 hari]

- ✓ **Convex Functions**
 - Definition: $f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$
 - Properties of convex functions
 - **BUATKAN:** Algoritma convexity check
 - **PSEUDOCODE:** Test convexity numerically
- ✓ **Convex Sets**
 - Extreme points
 - Intersection property
- X **SKIP:** Conjugate functions, subdifferentials

Active Set Method - [3 hari]

- ✓ **Active Set Strategy**
 - Working set concept
 - Adding/removing constraints
 - **BUATKAN:** Algoritma Active Set Method
 - **PSEUDOCODE:** Complete active set solver
- ✓ **Constraint Qualification**
 - Linear independence constraint qualification (LICQ)
- X **SKIP:** Sequential quadratic programming

Interior Point Methods - [2 hari]

- ✓ **Barrier Functions**
 - Logarithmic barrier
 - Central path
 - **BUATKAN:** Algoritma interior point (simplified)
 - **PSEUDOCODE:** Barrier method iteration
- X **SKIP:** Primal-dual interior point, path-following methods

Latihan Wajib:

- Hitung eigenvalues untuk 2 matriks 2x2
- Selesaikan 1 QP dengan active set (manual, 2 variabel)

- Trace barrier method untuk 1 masalah sederhana

Algoritma: Wolfe Method, Active Set, Interior Point

5. PEMROGRAMAN DINAMIKA DETERMINISTIK

💡 **LEARNING PATH:** Recursion → Stage-State → Bellman → DP Implementation

Teori Dasar yang Harus Dipelajari:

Rekursi - [2 hari]

- **Recursive Relations**
 - Base case dan recursive case
 - Recurrence equations
 - **BUATKAN:** Algoritma rekursi sederhana (Fibonacci)
 - **PSEUDOCODE:** Recursive function template
- **Recursion Tree**
 - Depth dan breadth
 - Overlapping subproblems
- **SKIP:** Master theorem, advanced recurrence solving

Prinsip Optimalitas (Bellman) - [2 hari]

- **Optimal Substructure**
 - Problem decomposition
 - Independent subproblems
- **Bellman's Principle**
 - "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision"
- **SKIP:** Hamilton-Jacobi-Bellman equation

Dynamic Programming Graph - [2 hari]

- **Stage-State Representation**
 - Directed Acyclic Graph (DAG)
 - State space definition
 - Decision variables
 - **BUATKAN:** Algoritma state space graph construction

- **PSEUDOCODE:** Build DP graph
- **Path Enumeration**
 - Complete enumeration
 - Pruning strategies
- **SKIP:** State aggregation

Forward/Backward Recursion - [3 hari]

- **Forward Recursion**
 - $f_k(s) = \min\{c_k(s,d) + f_{k+1}(s')\}$
 - Tabulation (bottom-up)
 - **BUATKAN:** Algoritma Forward DP
 - **PSEUDOCODE:** Forward recursion solver
- **Backward Recursion**
 - $f_k(s) = \min\{c_k(s,d) + f_{k-1}(s')\}$
 - Memoization (top-down)
 - **BUATKAN:** Algoritma Backward DP
 - **PSEUDOCODE:** Backward recursion with memoization
- **SKIP:** Approximate dynamic programming



Latihan Wajib:

- Selesaikan shortest path problem dengan DP (manual)
- Trace forward dan backward recursion untuk 1 masalah
- Buat state-transition diagram untuk knapsack problem

Algoritma: Forward Recursion, Backward Recursion, Bellman Equation

6. SISTEM ANTRIAN

💡 **LEARNING PATH:** Probability Basics → Distributions → Markov Chains → Queueing Models

Teori Dasar yang Harus Dipelajari:

Teori Probabilitas - [2 hari]

- **Variabel Random**
 - Discrete vs continuous
 - Probability mass/density function

- Expected value: $E[X] = \sum x \cdot P(x)$
- Variance: $Var(X) = E[X^2] - (E[X])^2$
- **BUATKAN:** Algoritma hitung expected value
- **PSEUDOCODE:** Calculate $E[X]$ and $Var(X)$
- **SKIP:** Moment generating functions

Distribusi Probabilitas - [3 hari]

- **Distribusi Poisson**
 - $P(X=k) = (\lambda^k e^{-\lambda})/k!$
 - Parameter λ (arrival rate)
 - **BUATKAN:** Algoritma generate Poisson random variable
 - **PSEUDOCODE:** Poisson probability calculation
- **Distribusi Eksponensial**
 - $f(x) = \lambda e^{-\lambda x}$
 - Memoryless property
 - Inter-arrival times
 - **BUATKAN:** Algoritma generate exponential random variable
 - **PSEUDOCODE:** Exponential PDF/CDF
- **SKIP:** Erlang distribution (kecuali perlu)

Proses Stokastik - [3 hari]

- **Markov Chains**
 - State transitions
 - Transition probability matrix P
 - Chapman-Kolmogorov equations
 - **BUATKAN:** Algoritma Markov chain simulation
 - **PSEUDOCODE:** State transition simulation
- **Steady-State Probabilities**
 - Balance equations: $\pi P = \pi$
 - Solving for π
 - **BUATKAN:** Algoritma solve steady-state
 - **PSEUDOCODE:** Steady-state solver

- **Birth-Death Process**
 - Birth rate λ , Death rate μ
 - Balance equations
- **SKIP:** Continuous-time Markov chains (CTMC)

Little's Law - [1 hari]

- **Fundamental Relationships**
 - $L = \lambda W$ (average number = arrival rate \times average time)
 - $L_q = \lambda W_q$ (for queue only)
 - $W = W_q + 1/\mu$
 - **BUATKAN:** Algoritma apply Little's Law
 - **PSEUDOCODE:** Calculate performance metrics

Queueing Models - [3 hari]

- **M/M/1 Queue**
 - $\rho = \lambda/\mu$ (utilization)
 - $L = \rho/(1-\rho)$
 - $W = 1/(\mu-\lambda)$
 - **BUATKAN:** Algoritma M/M/1 solver
 - **PSEUDOCODE:** Calculate all M/M/1 metrics
- **M/M/c Queue**
 - c servers
 - Erlang-C formula
 - **BUATKAN:** Algoritma M/M/c solver
 - **PSEUDOCODE:** Multi-server queue metrics
- **SKIP:** M/G/1, G/G/1 queues

Simulasi Monte Carlo - [2 hari]

- **Random Number Generation**
 - Uniform random numbers
 - Inverse transform method
 - **BUATKAN:** Algoritma simulasi antrian
 - **PSEUDOCODE:** Event-driven simulation

- **Statistical Analysis**
 - Sample mean, variance
 - Confidence intervals
- **SKIP:** Variance reduction techniques

 **Latihan Wajib:**

- Hitung $P(X=3)$ untuk Poisson($\lambda=2$)
- Selesaikan M/M/1 queue dengan $\lambda=3$, $\mu=5$
- Simulasi 20 customer arrivals manually

Algoritma: M/M/1, M/M/c, Monte Carlo Simulation

7. ANALISA JARINGAN

 **LEARNING PATH:** Graph Basics → Shortest Path → MST → Network Flow → Project Scheduling

Teori Dasar yang Harus Dipelajari:

Teori Graf Fundamental - [2 hari]

- **Representasi Graf**
 - Adjacency matrix ($n \times n$)
 - Adjacency list (linked list)
 - Edge list
 - **BUATKAN:** Algoritma konversi representasi graph
 - **PSEUDOCODE:** Matrix to list conversion
- **Jenis Graf**
 - Directed vs undirected
 - Weighted vs unweighted
 - Connected vs disconnected
- **SKIP:** Graph coloring, planarity

Path dan Cycle - [2 hari]

- **Path Properties**
 - Simple path vs cycle
 - Path weight/length
 - **BUATKAN:** Algoritma enumerate all paths (brute force)

- **PSEUDOCODE:** Path finding between two nodes
- **✗ SKIP:** Hamiltonian path, Eulerian path

Priority Queue & Heap - [2 hari]

- **✓ Heap Data Structure**
 - Min-heap, max-heap
 - Heap operations: insert, extract-min
 - **BUATKAN:** Algoritma heap operations
 - **PSEUDOCODE:** Min-heap implementation
- **✗ SKIP:** Fibonacci heap

Shortest Path Algorithms - [4 hari]

- **✓ Dijkstra's Algorithm**
 - Single-source shortest path
 - Non-negative weights
 - **BUATKAN:** Algoritma Dijkstra lengkap
 - **PSEUDOCODE:** Dijkstra with priority queue
- **✓ Bellman-Ford Algorithm**
 - Handles negative weights
 - Negative cycle detection
 - **BUATKAN:** Algoritma Bellman-Ford
 - **PSEUDOCODE:** Bellman-Ford with cycle check
- **✓ Floyd-Warshall Algorithm**
 - All-pairs shortest path
 - Dynamic programming approach
 - **BUATKAN:** Algoritma Floyd-Warshall
 - **PSEUDOCODE:** Floyd-Warshall with path reconstruction
- **✗ SKIP:** A* (heuristic search), Johnson's algorithm

Minimum Spanning Tree - [3 hari]

- **✓ MST Theory**
 - Cut property
 - Cycle property

- **Prim's Algorithm**
 - Greedy approach
 - Grow tree from single vertex
 - **BUATKAN:** Algoritma Prim
 - **PSEUDOCODE:** Prim with priority queue
- **Kruskal's Algorithm**
 - Sort edges by weight
 - Union-Find data structure
 - **BUATKAN:** Algoritma Kruskal
 - **PSEUDOCODE:** Kruskal with Union-Find
- **SKIP:** Borůvka's algorithm

Union-Find (Disjoint Set) - [2 hari]

- **Union-Find Structure**
 - Find operation
 - Union operation
 - Path compression
 - Union by rank
 - **BUATKAN:** Algoritma Union-Find
 - **PSEUDOCODE:** Union-Find with optimizations

Network Flow - [4 hari]

- **Flow Network Theory**
 - Capacity constraints
 - Flow conservation
 - Source and sink
- **Ford-Fulkerson Method**
 - Augmenting path
 - Residual graph
 - **BUATKAN:** Algoritma Ford-Fulkerson
 - **PSEUDOCODE:** Ford-Fulkerson framework
- **Edmonds-Karp Algorithm**

- BFS for finding augmenting path
- $O(VE^2)$ complexity
- **BUATKAN:** Algoritma Edmonds-Karp
- **PSEUDOCODE:** Edmonds-Karp with BFS
- **SKIP:** Push-relabel, Dinic's algorithm

Project Scheduling - [4 hari]

- **Activity Network**
 - AOA (Activity on Arc)
 - AON (Activity on Node)
 - Precedence relationships
 - **BUATKAN:** Algoritma construct AOA/AON
 - **PSEUDOCODE:** Build activity network
- **CPM (Critical Path Method)**
 - Forward pass: earliest start/finish (ES, EF)
 - Backward pass: latest start/finish (LS, LF)
 - Float/slack calculation: $\text{Float} = LS - ES$
 - Critical path identification
 - **BUATKAN:** Algoritma CPM lengkap
 - **PSEUDOCODE:** CPM with forward/backward pass
- **PERT (Program Evaluation Review Technique)**
 - Three time estimates: optimistic (a), most likely (m), pessimistic (b)
 - Expected time: $t_e = (a + 4m + b)/6$
 - Variance: $\sigma^2 = [(b-a)/6]^2$
 - Project completion probability
 - **BUATKAN:** Algoritma PERT
 - **PSEUDOCODE:** PERT with probabilistic analysis
- **SKIP:** Resource leveling, crashing

 **Latihan Wajib:**

- Trace Dijkstra untuk graph 6 nodes (manual)
- Hitung MST dengan Prim dan Kruskal
- Selesaikan max flow problem (5 nodes)

- Hitung critical path untuk project 8 activities

Algoritma: Dijkstra, Floyd-Warshall, Bellman-Ford, Prim, Kruskal, Ford-Fulkerson, Edmonds-Karp, CPM, PERT

8. LINE BALANCING

📍 **LEARNING PATH:** Production Concepts → Precedence Graph → Heuristics → Optimization

Teori Dasar yang Harus Dipelajari:

Production Theory - [2 hari]

- **Assembly Line Concepts**
 - Workstation
 - Cycle time (C)
 - Production rate = $1/C$
 - Theoretical minimum stations: $TM = \sum t_i / C$
 - **BUATKAN:** Algoritma calculate theoretical minimum
 - **PSEUDOCODE:** Compute TM and cycle time
- **Efficiency Measures**
 - Line efficiency = $(\sum t_i) / (n \times C) \times 100\%$
 - Balance delay = $1 - \text{Line efficiency}$
 - Smoothness index
- **SKIP:** Mixed-model assembly lines

Precedence Graph - [2 hari]

- **Directed Acyclic Graph (DAG)**
 - Task dependencies
 - Immediate predecessors
 - Topological ordering
 - **BUATKAN:** Algoritma topological sort
 - **PSEUDOCODE:** Topological ordering
- **Longest Path**
 - Positional weight
 - **BUATKAN:** Algoritma longest path in DAG
 - **PSEUDOCODE:** Calculate positional weights

- **✗ SKIP:** Transitive reduction

Heuristic Methods - [4 hari]

- **✓ Ranked Positional Weight (RPW)**
 - Calculate positional weight for each task
 - Rank tasks by positional weight
 - Assign tasks to stations
 - **BUATKAN:** Algoritma RPW lengkap
 - **PSEUDOCODE:** RPW heuristic
- **✓ Longest Operation Time (LOT)**
 - Rank by task duration
 - Greedy assignment
 - **BUATKAN:** Algoritma LOT
 - **PSEUDOCODE:** LOT heuristic
- **✓ Kilbridge-Wester Heuristic**
 - Column method
 - Zone-based allocation
 - **BUATKAN:** Algoritma Kilbridge-Wester
 - **PSEUDOCODE:** K-W column method
- **✗ SKIP:** COMSOAL, genetic algorithms for line balancing

Constraint Handling - [1 hari]

- **✓ Feasibility Checks**
 - Precedence constraint satisfaction
 - Cycle time constraint
 - **BUATKAN:** Algoritma feasibility check
 - **PSEUDOCODE:** Validate station assignment

Latihan Wajib:

- Hitung positional weight untuk 8 tasks
- Balance line dengan RPW (manual)
- Compare LOT vs RPW untuk 1 masalah
- Hitung line efficiency dan balance delay

Algoritma: Ranked Positional Weight, Longest Operation Time, Kilbridge-Wester

9. TEORI KEPUTUSAN

📍 **LEARNING PATH:** Probability → Decision Criteria → Tree Analysis → Bayesian Methods

Teori Dasar yang Harus Dipelajari:

Decision Theory Fundamentals - [2 hari]

- **Decision Elements**
 - Decision alternatives (actions)
 - States of nature (events)
 - Payoff/outcome matrix
 - **BUATKAN:** Algoritma construct payoff matrix
 - **PSEUDOCODE:** Decision matrix representation
- **Decision Environment**
 - Certainty
 - Risk (probabilistic)
 - Uncertainty (no probabilities)
- **SKIP:** Fuzzy decision theory

Probability Theory - [2 hari]

- **Basic Probability**
 - Joint probability: $P(A,B)$
 - Conditional probability: $P(A|B)$
 - Bayes' theorem: $P(A|B) = P(B|A)P(A)/P(B)$
 - **BUATKAN:** Algoritma Bayes' theorem calculation
 - **PSEUDOCODE:** Apply Bayes' rule
- **Expected Value**
 - $E[X] = \sum x_i \cdot P(x_i)$
 - Expected Monetary Value (EMV)
 - **BUATKAN:** Algoritma calculate EMV
 - **PSEUDOCODE:** EMV for each alternative
- **SKIP:** Subjective probability elicitation

Decision Criteria under Uncertainty - [3 hari]

- **Minimax (Conservative)**
 - Minimize maximum loss
 - Maximin: maximize minimum gain
 - **BUATKAN:** Algoritma Minimax
 - **PSEUDOCODE:** Minimax/Maximin decision
- **Expected Value Criterion**
 - Choose alternative with highest EMV
 - **BUATKAN:** Algoritma EMV selection
 - **PSEUDOCODE:** EMV-based decision
- **Regret/Opportunity Loss**
 - Minimax regret criterion
 - **BUATKAN:** Algoritma minimax regret
 - **PSEUDOCODE:** Calculate regret matrix
- **SKIP:** Hurwicz criterion, Laplace criterion

Decision Tree Analysis - [4 hari]

- **Tree Structure**
 - Decision nodes (\square)
 - Chance nodes (\circ)
 - Terminal nodes (\triangle)
 - **BUATKAN:** Algoritma construct decision tree
 - **PSEUDOCODE:** Build tree structure
- **Backward Induction (Rollback)**
 - Start from terminal nodes
 - Calculate expected values
 - Propagate backwards
 - **BUATKAN:** Algoritma backward induction
 - **PSEUDOCODE:** Rollback algorithm
- **Expected Value of Perfect Information (EVPI)**
 - $EVPI = EVwPI - EMV$

- Maximum willing to pay for information
- **BUATKAN:** Algoritma calculate EVPI
- **PSEUDOCODE:** EVPI computation
- **✗ SKIP:** Influence diagrams

Bayesian Decision Theory - [3 hari]

- **✓ Prior and Posterior Probabilities**
 - Prior: $P(\theta)$
 - Likelihood: $P(\text{data}|\theta)$
 - Posterior: $P(\theta|\text{data})$
 - **BUATKAN:** Algoritma Bayesian updating
 - **PSEUDOCODE:** Posterior calculation
- **✓ Expected Value of Sample Information (EVSI)**
 - Value of imperfect information
 - $\text{EVSI} = \text{EVwSI} - \text{EMV}$
 - **BUATKAN:** Algoritma calculate EVSI
 - **PSEUDOCODE:** EVSI computation
- **✗ SKIP:** Sequential decision analysis

Utility Theory (Overview) - [1 hari]

- **✓ Utility Function**
 - Risk averse, risk neutral, risk seeking
 - Expected utility vs expected value
- **✗ SKIP:** Von Neumann-Morgenstern axioms

Latihan Wajib:

- Buat payoff matrix untuk 3 alternatives \times 4 states
- Hitung minimax dan EMV
- Construct dan solve decision tree (3 levels)
- Apply Bayes' theorem untuk updating probabilities

Algoritma: Minimax/Maximin, EMV, Decision Tree Analysis, Bayesian Decision

E. DELIVERABLES TEORI

Setiap kelompok WAJIB menyertakan dalam laporan:

1. **Bab Teori Dasar** (minimal 5-7 halaman):

- Penjelasan lengkap konsep matematika/struktur data yang digunakan
- Ilustrasi dan diagram
- Contoh perhitungan manual
- Referensi yang kredibel

2. **Mind Map/Concept Map:**

- Hubungan antara teori dasar dengan algoritma
- Visualisasi dependensi konsep

3. **Algoritma dan Pseudocode:**

- Untuk setiap konsep teori yang ditandai "BUATKAN"
- Flowchart algoritma utama
- Pseudocode yang jelas dan terstruktur

4. **Quiz/Self-Assessment:**

- 10 soal untuk menguji pemahaman teori dasar
- Dengan kunci jawaban dan penjelasan

F. STUDI KASUS UNTUK SETIAP ALGORITMA

Setiap algoritma memiliki studi kasus dunia nyata atau aplikasi terkini:

1. Pemrograman Linear

Metode Grafik/Simpleks:

- **Studi Kasus:** Optimasi produksi startup manufaktur Indonesia (mis: produksi UMKM)
- **Dataset:** Data produksi, biaya bahan baku, kapasitas mesin
- **Aplikasi:** Maksimasi profit dengan resource terbatas

Dualitas:

- **Studi Kasus:** Pricing strategy untuk perusahaan logistik (JNE, JNT, SiCepat)
- **Aplikasi:** Shadow prices untuk decision making

2. Pemrograman Bilangan Bulat

Branch and Bound / Cutting Plane:

- **Studi Kasus:** Penjadwalan shift perawat di rumah sakit (trending post-pandemic)

- **Dataset:** Jumlah perawat, shift requirements, preferensi
- **Aplikasi:** Minimize cost while meeting coverage

Alternative:

- **Studi Kasus:** Facility location problem untuk dark kitchen/cloud kitchen
 - **Aplikasi:** Optimal location untuk minimize delivery time
-

3. Pemrograman Tak Linear

Gradient Descent / Newton-Raphson:

- **Studi Kasus:** Optimasi portofolio investasi (saham, crypto, obligasi)
- **Dataset:** Historical returns, risks, correlations
- **Aplikasi:** Maximize return dengan given risk tolerance

Lagrange / KKT:

- **Studi Kasus:** Resource allocation untuk startup tech (budget, manpower)
 - **Aplikasi:** Optimal allocation dengan multiple constraints
-

4. Pemrograman Kuadratik

Active Set / Interior Point:

- **Studi Kasus:** Energy management untuk smart grid/microgrid
- **Dataset:** Energy generation, demand, storage capacity
- **Aplikasi:** Minimize cost dengan quadratic losses

Alternative:

- **Studi Kasus:** Traffic flow optimization (Jakarta smart city project)
 - **Aplikasi:** Minimize congestion dengan quadratic cost
-

5. Pemrograman Dinamika

Forward/Backward Recursion:

- **Studi Kasus:** Inventory management untuk e-commerce (Tokopedia, Shopee)
- **Dataset:** Demand forecast, holding cost, ordering cost
- **Aplikasi:** Dynamic lot-sizing problem

Alternative:

- **Studi Kasus:** Equipment replacement policy untuk perusahaan transportasi

- **Aplikasi:** When to replace aging fleet (bus, truck)
-

6. Sistem Antrian

M/M/1 / M/M/c:

- **Studi Kasus:** Customer service center optimization (call center bank, e-wallet)
- **Dataset:** Arrival rate, service time, number of agents
- **Aplikasi:** Staffing decisions untuk minimize wait time

Monte Carlo Simulation:

- **Studi Kasus:** Drive-thru queue di fast food chain (McDonald's, KFC)
 - **Dataset:** Historical arrival data, service times
 - **Aplikasi:** Optimize number of service windows
-

7. Analisa Jaringan

Dijkstra / Bellman-Ford:

- **Studi Kasus:** Routing optimization untuk ride-hailing (Gojek, Grab)
- **Dataset:** Jakarta road network, traffic conditions
- **Aplikasi:** Fastest route dengan real-time traffic

MST (Prim/Kruskal):

- **Studi Kasus:** Fiber optic network design untuk ISP
- **Dataset:** Cities, distances, installation costs
- **Aplikasi:** Minimize total cost of network infrastructure

Max Flow:

- **Studi Kasus:** Supply chain optimization untuk FMCG distribution
- **Dataset:** Warehouses, retailers, capacities
- **Aplikasi:** Maximum throughput dari factory ke outlets

CPM/PERT:

- **Studi Kasus:** Project management untuk software development (Agile sprint)
 - **Dataset:** Tasks, dependencies, duration estimates
 - **Aplikasi:** Critical path dan project duration estimation
-

8. Line Balancing

RPW / LOT / Kilbridge-Wester:

- **Studi Kasus:** Assembly line balancing untuk automotive assembly plant
- **Dataset:** Tasks, precedence, task times
- **Aplikasi:** Minimize number of workstations

Alternative:

- **Studi Kasus:** PCB assembly line untuk electronics manufacturing
 - **Aplikasi:** Balance untuk achieve target cycle time
-

9. Teori Keputusan

Minimax / EMV:

- **Studi Kasus:** Investment decision dalam ketidakpastian (property, stocks)
- **Dataset:** Economic scenarios, probabilities, payoffs
- **Aplikasi:** Risk-based decision making

Decision Tree:

- **Studi Kasus:** New product launch decision untuk startup
- **Dataset:** Market research, probabilities, costs, revenues
- **Aplikasi:** Go/No-go decision dengan sequential decisions

Bayesian Decision:

- **Studi Kasus:** Medical diagnosis decision support system
 - **Dataset:** Symptoms, test results, disease probabilities
 - **Aplikasi:** Treatment decision dengan updated probabilities
-

G. TUGAS KELOMPOK

Setiap kelompok akan:

1. **Mempelajari** algoritma yang dipilih secara mendalam
2. **Memahami teori dasar** yang membentuk algoritma
3. **Mengimplementasikan** algoritma dalam bahasa pemrograman (Python/MATLAB/Java/C++)
4. **Mengerjakan studi kasus** yang telah ditentukan di atas
5. **Menyusun laporan** yang berisi:
 - Teori dan konsep algoritma
 - Algoritma dan pseudocode untuk konsep-konsep teori

- Flowchart
- Implementasi kode program dengan dataset nyata
- Analisis hasil dan kompleksitas
- Pembahasan relevansi dengan isu terkini
- Kesimpulan dan saran

6. **Mempresentasikan** hasil kerja kelompok di kelas

H. DELIVERABLES

1. **Laporan tertulis** (format PDF)
 2. **Source code** program (dengan dokumentasi)
 3. **Slide presentasi** (format PPT/PDF)
 4. **Video demo** program (opsional, nilai bonus)
-

I. TIMELINE & JADWAL PROYEK 10 PERTEMUAN

Proyek ini berlangsung selama **10 pertemuan** dengan output spesifik di setiap pertemuan:

PERTEMUAN 1: Kick-Off & Pembentukan Kelompok

Agenda:

- Penjelasan instruksi tugas oleh dosen (30 menit)
- Pembentukan kelompok (20 menit)
- Pemilihan algoritma (30 menit)
- Diskusi studi kasus awal (20 menit)

Output yang Harus Diserahkan:

 **Dokumen: Proposal Awal (1-2 halaman)**

- Nama kelompok dan anggota (NPM, email, no. HP)
- Algoritma yang dipilih
- Alasan pemilihan algoritma
- Studi kasus yang akan dikerjakan (overview)
- Pembagian tugas awal antar anggota

Format Pengumpulan: PDF via email/LMS

Penilaian: ✓ Complete / X Incomplete (tidak ada nilai numerik)

PERTEMUAN 2: Pendalaman Teori Dasar - Part 1

Agenda:

- Mahasiswa belajar mandiri teori dasar (konsep fundamental)
- Konsultasi kelompok dengan dosen (15 menit per kelompok)

Output yang Harus Diserahkan:

Dokumen: Mind Map Teori Dasar

- Mind map yang menunjukkan hubungan antar konsep
- Minimal 8-12 node konsep
- Gunakan tools: Miro, XMind, atau tulis tangan (difoto)
- Tandai konsep yang sudah dipahami vs belum

Catatan Pembelajaran:

- Daftar konsep yang dipelajari minggu ini
- Sumber referensi yang digunakan (min. 3 sumber)
- Kesulitan yang dihadapi
- Pertanyaan untuk dosen (min. 3 pertanyaan)

Format Pengumpulan: PDF (Mind Map + Catatan)

Penilaian: 5% dari nilai akhir

PERTEMUAN 3: Pendalaman Teori Dasar - Part 2

Agenda:

- Checkpoint 1: Presentasi informal teori (5 menit per kelompok)
- Tanya jawab dengan dosen
- Feedback dan arahan perbaikan

Output yang Harus Diserahkan:

Slide Presentasi Teori (Max 10 slide)

- Slide 1: Algoritma & anggota kelompok
- Slide 2-7: Penjelasan 5-6 konsep teori utama
- Slide 8-9: Contoh perhitungan manual (minimal 1)
- Slide 10: Rencana implementasi

Hand Calculation - Draft Awal (3 soal)

- Kerjakan 3 soal latihan manual dari bagian "Latihan Wajib"

- Tunjukkan semua langkah perhitungan
- Boleh tulis tangan (difoto dengan jelas)

Format Pengumpulan: PPT/PDF + PDF (Hand Calculation)

Penilaian: 10% dari nilai akhir

Catatan: Presentasi dinilai dari:

- Kejelasan penjelasan (40%)
 - Kemampuan menjawab pertanyaan (40%)
 - Visualisasi & contoh (20%)
-

PERTEMUAN 4: Penyelesaian Teori & Mulai Design

Agenda:

- Finalisasi pemahaman teori
- Mulai mendesain arsitektur sistem
- Konsultasi dataset dan tools

Output yang Harus Diserahkan:



Struktur:

1. Pendahuluan teori (0.5 hal)
2. Konsep fundamental (2-3 hal)
 - Definisi lengkap
 - Formula matematika
 - Ilustrasi/diagram
3. Konsep lanjutan (2-3 hal)
 - Penjelasan mendalam
 - Hubungan antar konsep
4. Contoh perhitungan manual (1-1.5 hal)
 - Minimal 2 contoh berbeda
 - Langkah-langkah detail
5. Daftar pustaka (min. 5 referensi)



- Flowchart algoritma utama

- Arsitektur sistem (diagram blok)
- Struktur data yang akan digunakan
- Pseudocode algoritma (untuk konsep-konsep yang ditandai "BUATKAN")
- Rencana dataset (sumber, jumlah data, format)

Hand Calculation - COMPLETE (10 soal)

- 10 soal latihan manual selesai semua (dari "Latihan Wajib")
- Rapi, jelas, dan benar
- Include penjelasan di setiap langkah

Format Pengumpulan: PDF (Teori) + PDF (Design) + PDF (Hand Calc)

Penilaian: 15% dari nilai akhir

PERTEMUAN 5: Implementasi Fase 1 - Setup & Basic

Agenda:

- Mulai coding
- Setup environment & library
- Implementasi struktur dasar

Output yang Harus Diserahkan:

Source Code - Version 1.0

- Repository GitHub (harus public atau invite dosen)
- README.md lengkap:
 - Cara instalasi
 - Dependencies
 - Cara menjalankan
 - Status pengembangan
- Code struktur dasar:
 - Import library
 - Load dataset
 - Fungsi-fungsi helper
 - Struktur data utama

Dataset Preliminary Analysis:

- Dataset yang akan digunakan (upload sample 100 data)

- Statistik deskriptif (ukuran, fitur, distribusi)
- Data preprocessing yang direncanakan
- Visualisasi awal (3-5 plot)

Development Log:

- Progress minggu ini (apa yang sudah dikerjakan)
- Pembagian tugas (siapa mengerjakan apa)
- Challenges & solutions
- Rencana minggu depan

Format Pengumpulan:

- Link GitHub Repository
- PDF (Dataset Analysis + Development Log)

Penilaian: 10% dari nilai akhir

PERTEMUAN 6: Implementasi Fase 2 - Core Algorithm

Agenda:

- Checkpoint 2: Code review awal
- Implementasi algoritma utama
- Testing dengan data kecil

Output yang Harus Diserahkan:

Source Code - Version 2.0

- Algoritma utama sudah terimplementasi (60-70% selesai)
- Code sudah di-comment (dokumentasi inline)
- Unit testing untuk fungsi-fungsi kunci
- Git commit history yang jelas

Testing Report - Preliminary:

- Test dengan dataset kecil (50-100 data)
- Output algoritma (screenshot/log)
- Validasi manual vs program (apakah output benar?)
- Bug yang ditemukan dan cara fixing

Video Progress Demo (3-5 menit):

- Rekam screen saat menjalankan program

- Narasi menjelaskan setiap step
- Tunjukkan output yang dihasilkan
- Jelaskan bagian yang masih belum selesai

Format Pengumpulan:

- Link GitHub (update)
- PDF (Testing Report)
- Link Video (YouTube/Drive - unlisted)

Penilaian: 10% dari nilai akhir

PERTEMUAN 7: Implementasi Fase 3 - Integration & Optimization

Agenda:

- Integrasi semua komponen
- Optimasi performa
- Testing dengan full dataset

Output yang Harus Diserahkan:

 **Source Code - Version 3.0 (COMPLETE)**

- Semua fitur terimplementasi (100%)
- Code clean dan terorganisir
- Dokumentasi lengkap (docstring)
- Error handling
- Visualization output (grafik, tabel, dll)

 **Experimental Results:**

- Testing dengan full dataset
- Performance metrics:
 - Objective function value
 - Computation time
 - Number of iterations
 - Solution quality
 - Comparison dengan theoretical optimal (jika ada)
- Perbandingan dengan baseline/ekspektasi
- Analisis error (jika ada)



Grafik & Visualisasi (min. 5 visualisasi):

- Grafik convergence (untuk iterative algorithms)
- Solution visualization (untuk spatial problems)
- Sensitivity analysis plots
- Performance comparison charts
- Parameter tuning results

Format Pengumpulan:

- Link GitHub (final code)
- PDF (Experimental Results dengan visualisasi)

Penilaian: 15% dari nilai akhir

PERTEMUAN 8: Finalisasi & Laporan

Agenda:

- Checkpoint 3: Final code review & viva voce
- Penulisan laporan akhir
- Persiapan presentasi

Output yang Harus Diserahkan:



LAPORAN AKHIR LENGKAP (20-30 halaman)

Struktur Laporan:

1. **Cover & Identitas Kelompok** (1 hal)
2. **Abstrak** (0.5 hal)
3. **BAB I: Pendahuluan** (2-3 hal)
 - Latar belakang
 - Rumusan masalah
 - Tujuan
 - Manfaat
4. **BAB II: Teori Dasar** (5-7 hal)
 - Konsep fundamental
 - Konsep algoritma
 - Penelitian terkait (literature review)
5. **BAB III: Metodologi** (3-4 hal)

- Flowchart
- Pseudocode (untuk konsep-konsep teori)
- Dataset description
- Tools & library
- Rancangan sistem

6. **BAB IV: Implementasi (4-6 hal)**

- Penjelasan kode
- Screenshot kode penting
- Struktur program
- Challenges & solutions

7. **BAB V: Hasil & Analisis (4-6 hal)**

- Experimental setup
- Results & metrics
- Visualisasi
- Analisis mendalam
- Perbandingan dengan teori
- Sensitivity analysis

8. **BAB VI: Kesimpulan & Saran (1-2 hal)**

- Kesimpulan
- Keterbatasan
- Saran pengembangan

9. **Daftar Pustaka (min. 10 referensi)**

10. **Lampiran** (kode lengkap, dataset sample)

 **Slide Presentasi Final (15-20 slide)**

- Template profesional
- Visualisasi menarik
- Fokus pada hasil & demo

 **Package Lengkap:**

- Source code (ZIP + GitHub link)
- Dataset (sample jika terlalu besar)
- Video demo program (5-7 menit)

- Dokumentasi cara penggunaan (User Manual)

Format Pengumpulan:

- PDF (Laporan) - upload ke LMS
- PPT/PDF (Slide) - upload ke LMS
- ZIP (Code + Dataset) - upload atau Google Drive link
- Link Video Demo

Penilaian: 15% dari nilai akhir

Catatan: Laporan akan dicek plagiarism dengan Turnitin (max 20% similarity)

PERTEMUAN 9: PRESENTASI KELOMPOK 1

Agenda:

- Presentasi 50% kelompok (diundi)
- Setiap kelompok: 15 menit presentasi + 10 menit tanya jawab
- Format: Offline/Hybrid

Yang Harus Dipersiapkan:



Presentasi (15 menit):

Struktur Presentasi:

1. Opening (1 menit)

- Salam, perkenalan kelompok
- Outline presentasi

2. Teori Dasar (3 menit)

- Konsep fundamental (pilih 2-3 konsep paling penting)
- Kenapa konsep ini penting untuk algoritma?
- Contoh sederhana (1 saja, yang paling ilustratif)

3. Studi Kasus & Dataset (2 menit)

- Problem yang diselesaikan
- Dataset yang digunakan
- Relevansi dengan dunia nyata

4. Implementasi (3 menit)

- Flowchart/arsitektur sistem
- Penjelasan algoritma secara high-level

- Highlight kode penting (jangan semua kode!)

5. Demo Live (4 menit)

- Jalankan program secara live
- Tunjukkan input → proses → output
- Minimal 2-3 test cases
- **PENTING: Siapkan video backup jika demo gagal!**

6. Hasil & Analisis (3 menit)

- Performance metrics
- Visualisasi hasil
- Perbandingan dengan ekspektasi/baseline
- Insights yang didapat

7. Kesimpulan (1 menit)

- Pembelajaran utama
- Keterbatasan
- Saran pengembangan

💡 Tanya Jawab (10 menit):

- Dosen akan bertanya 5-7 pertanyaan
- Semua anggota kelompok HARUS menjawab minimal 1 pertanyaan
- Jawaban harus berdasarkan pemahaman, bukan hafalan

Yang Dinilai:

- Pemahaman teori dasar (20%)
- Kualitas implementasi & demo (30%)
- Analisis hasil (20%)
- Kemampuan presentasi (15%)
- Kemampuan menjawab pertanyaan (15%)

Tips Sukses:

- Latihan presentasi minimal 3x sebelum hari H
- Time management ketat (jangan over/under time)
- Siapkan backup plan (video demo, screenshot)
- Semua anggota harus berbicara (bagi porsi merata)
- Kontak mata dengan dosen & audiens

- Antusias dan percaya diri!

Format Pengumpulan:

- Slide presentasi (upload sebelum presentasi)
- Laptop + proyektor siap
- Backup file di USB/cloud

Penilaian: 10% dari nilai akhir (Presentasi)

PERTEMUAN 10: PRESENTASI KELOMPOK 2

Agenda:

- Presentasi 50% kelompok sisanya
- Format sama dengan Pertemuan 9

Yang Harus Dipersiapkan:

- Sama seperti Pertemuan 9
- Kelompok yang presentasi di pertemuan 9 harus hadir sebagai audiens
- Audiens bisa memberikan pertanyaan tambahan (bonus point)

Output yang Harus Diserahkan:

- Slide presentasi final
- Demo live program

Penilaian: 10% dari nilai akhir (Presentasi)

I.2. REKAPITULASI PENILAIAN

Pertemuan	Deliverable	Bobot
1	Proposal Awal	-
2	Mind Map + Catatan	5%
3	Slide Teori + Hand Calc Draft	10%
4	Teori Final + Design + Hand Calc Complete	15%
5	Code v1.0 + Dataset Analysis	10%
6	Code v2.0 + Testing + Video	10%
7	Code v3.0 + Experimental Results	15%

Pertemuan	Deliverable	Bobot
8	Laporan Final + Slide + Package	15%
9-10	Presentasi + Demo + Q&A	20%
TOTAL		100%

I.3. ATURAN PENGUMPULAN

1. **Deadline ketat:** Lewat deadline = potong 10% per hari (max 3 hari)
 2. **Format harus sesuai:** PDF untuk dokumen, PPT untuk slide, ZIP untuk code
 3. **Naming convention:** Kelompok[No]Pertemuan[No][NamaFile].pdf
 - Contoh: Kelompok01_Pertemuan03_SlideTeori.pdf
 4. **Upload ke LMS:** Semua file upload ke LMS (kecuali code ke GitHub)
 5. **Backup:** Simpan semua file di Google Drive kelompok (just in case)
-

I.4. KONSULTASI TAMBAHAN

Dosen menyediakan **office hours** untuk konsultasi:

- **Hari:** [Tentukan hari]
- **Waktu:** [Tentukan waktu]
- **Tempat:** [Tentukan tempat/link Zoom]
- **Durasi:** 15-20 menit per kelompok
- **Booking:** Via email minimal H-1

Sangat direkomendasikan untuk konsultasi di:

- Pertemuan 2-3 (validasi pemahaman teori)
 - Pertemuan 5-6 (bantuan debugging)
 - Pertemuan 7-8 (feedback laporan)
-

I.5. PENALTY & BONUS

Penalty:

- ✖ **Tidak hadir konsultasi wajib (checkpoint):** -10%
- ✖ **Plagiarisme (similarity > 30%):** Nilai E (tidak lulus)
- ✖ **Copy-paste code tanpa pemahaman:** Nilai maksimal C

- ✖ **Keterlambatan pengumpulan:** -10% per hari (max 3 hari)
- ✖ **Anggota tidak kontribusi** (berdasarkan Git history): Nilai individu dikurangi 20-50%
- ✖ **Hand calculation tidak lengkap/salah:** Potong 5-10% dari deliverable terkait

Bonus:

- ✿ **GitHub repo rapi + dokumentasi excellent:** +5%
- ✿ **Implementasi fitur tambahan beyond requirement:** +5%
- ✿ **Publikasi medium/blog tentang project:** +3%
- ✿ **Video tutorial penggunaan program (YouTube):** +3%
- ✿ **Audiens bertanya saat presentasi kelompok lain** (max 3 pertanyaan): +1% per pertanyaan
- ✿ **Perbandingan dengan algoritma lain (benchmarking):** +3%
- ✿ **Interactive visualization/dashboard:** +3%

Max bonus: +10% (tidak bisa melebihi 100)

J. KRITERIA PENILAIAN DETAIL

1. Pemahaman Teori Dasar (20%)

- Kelengkapan konsep teori yang dipelajari
- Kedalaman pemahaman (bukan hafalan)
- Kemampuan menjelaskan dengan kata-kata sendiri
- Kualitas contoh perhitungan manual
- Mind map dan concept map

2. Pemahaman Algoritma (20%)

- Pemahaman cara kerja algoritma
- Analisis kompleksitas (time & space)
- Kelebihan dan kekurangan algoritma
- Kondisi optimal penggunaan
- Hubungan teori dengan implementasi

3. Kualitas Implementasi Kode (25%)

- **Correctness (40%):**
 - Algoritma berjalan dengan benar
 - Output sesuai ekspektasi

- Handle edge cases
- **Code Quality (30%):**
 - Clean code (readable, maintainable)
 - Proper naming conventions
 - Modular structure
 - Error handling
- **Documentation (20%):**
 - Inline comments
 - Docstrings
 - README.md lengkap
 - User manual
- **Testing (10%):**
 - Unit tests
 - Integration tests
 - Test cases coverage

4. Analisis dan Pembahasan (15%)

- Analisis hasil eksperimen
- Visualisasi yang informatif
- Perbandingan dengan teori/benchmark
- Sensitivity analysis
- Interpretasi insights
- Diskusi keterbatasan

5. Laporan Tertulis (10%)

- Struktur laporan yang baik
- Bahasa Indonesia yang baik dan benar
- Format dan tata letak profesional
- Kelengkapan referensi (min 10 sumber)
- Originalitas (max 20% similarity)

6. Presentasi (10%)

- Kejelasan penyampaian
- Time management

- Kualitas slide dan visualisasi
 - Demo yang meyakinkan
 - Team coordination
 - Kemampuan menjawab pertanyaan
 - Antusiasme
-

K. TOOLS DAN BAHASA PEMROGRAMAN

Bahasa Pemrograman yang Direkomendasikan:

1. **Python** (Highly Recommended)
 - Libraries: NumPy, SciPy, PuLP, OR-Tools, NetworkX, Matplotlib
 - Jupyter Notebook untuk dokumentasi interaktif
2. **MATLAB/Octave**
 - Built-in optimization toolboxes
 - Good untuk rapid prototyping
3. **Java/C++**
 - Untuk performance-critical applications
 - Object-oriented design

Libraries untuk Operations Research:

Python:

- numpy, scipy - Mathematical operations
- pulp, cvxpy - Linear/Convex optimization
- networkx - Graph algorithms
- simpy - Discrete event simulation (queueing)
- matplotlib, seaborn, plotly - Visualization
- pandas - Data manipulation

MATLAB:

- Optimization Toolbox
- Statistics and Machine Learning Toolbox

Development Tools:

- **IDE:** VSCode, PyCharm, Jupyter Notebook
- **Version Control:** Git + GitHub

- **Visualization:** Matplotlib, Plotly, D3.js
 - **Documentation:** Sphinx, MkDocs
 - **Testing:** pytest, unittest
-

L. SUMBER BELAJAR TAMBAHAN

Buku Referensi:

1. "**Introduction to Operations Research**" - Hillier & Lieberman
 - Comprehensive OR textbook
 - Bab yang relevan untuk setiap algoritma
2. "**Operations Research: An Introduction**" - Taha
 - Detailed examples and exercises
 - Good for self-study
3. "**Integer Programming**" - Wolsey
 - For integer programming algorithms
4. "**Nonlinear Programming**" - Bertsekas
 - For nonlinear optimization
5. "**Introduction to Algorithms**" - Cormen et al.
 - For graph algorithms

Online Resources:

Video Tutorials:

- MIT OpenCourseWare - Operations Research
- Stanford Online - Optimization courses
- YouTube channels: Abdul Bari, William Fiset

Interactive Learning:

- GeeksforGeeks - Algorithm tutorials
- VisuAlgo - Algorithm visualization
- Khan Academy - Math fundamentals

Documentation:

- SciPy documentation
- OR-Tools documentation
- NetworkX documentation

Papers & Articles:

- Google Scholar untuk research papers
 - arXiv untuk latest research
 - Medium/Towards Data Science untuk practical tutorials
-

M. CONTOH STRUKTUR CODE

Struktur Folder yang Direkomendasikan:

```
project-root/
|
├── README.md
├── requirements.txt
└── .gitignore
|
├── data/
│   ├── raw/      # Original data
│   ├── processed/ # Processed data
│   └── sample/   # Sample data for testing
|
└── src/
    ├── __init__.py
    ├── algorithm.py  # Core algorithm implementation
    ├── utils.py      # Helper functions
    ├── visualization.py # Plotting functions
    └── preprocessing.py # Data preprocessing
|
└── tests/
    ├── test_algorithm.py
    └── test_utils.py
|
└── notebooks/
```

```
|   |-- 01_data_exploration.ipynb  
|   |-- 02_algorithm_development.ipynb  
|   |-- 03_results_analysis.ipynb  
|  
|  
└── results/  
    ├── figures/      # Generated plots  
    ├── logs/        # Execution logs  
    └── outputs/     # Algorithm outputs  
|  
└── docs/  
    ├── user_manual.md  
    └── technical_report.pdf
```

Template README.md:

markdown

[Nama Algoritma] - Riset Operasi

Deskripsi

Brief description of the algorithm and problem being solved.

Team Members

- Nama 1 (NPM) - Role
- Nama 2 (NPM) - Role
- Nama 3 (NPM) - Role

Installation

Prerequisites

- Python 3.8+
- pip

Setup

```
```bash
pip install -r requirements.txt
```

## Usage

### Running the Algorithm

```bash
python src/algorithm.py --input data/sample/input.csv
```

### Running Tests

```bash
pytest tests/
```

## Dataset

Description of the dataset used, source, and format.

## Results

Summary of key findings and performance metrics.

## License

[Your License]
```

N. TIPS SUKSES

Untuk Pembelajaran Teori:

1. Mulai dari konsep paling dasar, jangan skip
2. Buat catatan/ringkasan dengan tangan (lebih mudah diingat)
3. Kerjakan latihan manual sebelum coding
4. Gunakan visualisasi untuk memahami konsep abstrak

5. Diskusi dengan anggota kelompok (peer learning)

Untuk Implementasi:

1. Test dengan data kecil dulu sebelum full dataset
2. Commit ke Git secara regular (jangan tunggu selesai)
3. Tulis dokumentasi sambil coding (jangan di akhir)
4. Debug dengan print statements atau debugger
5. Validasi output dengan perhitungan manual

Untuk Laporan:

1. Mulai menulis sejak awal, jangan tunggu akhir
2. Gunakan template yang sudah diberikan
3. Pastikan semua gambar/tabel ada caption dan dijelaskan
4. Cite semua sumber yang digunakan
5. Proofread minimal 2x sebelum submit

Untuk Presentasi:

1. Rehearsal minimal 3x
2. Time your presentation (use timer)
3. Prepare for common questions
4. Have backup plan (video, screenshots)
5. Make eye contact, speak clearly

Time Management:

- Week 1-2: Theory learning (40% effort)
- Week 3-4: Design & setup (20% effort)
- Week 5-7: Implementation (30% effort)
- Week 8-10: Testing, documentation, presentation (10% effort)

O. FREQUENTLY ASKED QUESTIONS (FAQ)

Q1: Bolehkah menggunakan library/solver existing (seperti PuLP, CPLEX)?

A: Boleh untuk validasi, tetapi Anda HARUS implementasi algoritma dari scratch untuk pemahaman. Bisa bandingkan hasil Anda dengan library.

Q2: Bagaimana jika algoritma tidak converge atau hasil tidak optimal?

A: Dokumentasikan hasil apa adanya. Jelaskan why, analisis penyebabnya, dan diskusikan di laporan. Ini bagian dari learning process.

Q3: Dataset harus real atau boleh synthetic?

A: Prioritas: real data. Jika tidak tersedia, boleh synthetic tapi harus realistic (mimic real-world characteristics).

Q4: Berapa ukuran dataset minimum?

A: Minimal 100 data points untuk demonstrasi. Lebih besar lebih baik untuk analisis yang meaningful.

Q5: Boleh ganti algoritma setelah dikonfirmasi?

A: Tidak boleh, kecuali ada alasan kuat dan persetujuan dosen.

Q6: Apakah semua anggota harus coding?

A: Ya. Distribusi tugas harus merata. Git history akan dicek untuk memastikan kontribusi setiap anggota.

Q7: Bagaimana cara validasi bahwa implementasi benar?

A:

- Hitung manual untuk kasus kecil
 - Compare dengan hasil library (jika ada)
 - Test dengan known benchmarks
 - Cek apakah satisfy optimality conditions
-

Q8: Laporan harus bahasa Indonesia atau Inggris?

A: Bahasa Indonesia untuk laporan. Bahasa Inggris untuk code/comments (best practice).

Q9: Bagaimana jika dataset terlalu besar untuk di-upload?

A: Upload sample (100-1000 rows) dan provide link ke full dataset (Google Drive/Kaggle).

Q10: Harus mengerjakan SEMUA teori yang tercantum?

A: Tidak. Focus pada teori yang relevan dengan algoritma pilihan Anda. Lihat bagian masing-masing algoritma.

P. KONTAK & SUPPORT

Dosen Pengampu:

- **Nama:** [Nama Dosen]
- **Email:** [Email Dosen]
- **Office:** [Ruang]
- **Phone/WA:** [Nomor]

Asisten/Teaching Assistant:

- **Nama:** [Nama Asisten]
- **Email:** [Email]
- **WA Group:** [Link]

Jam Konsultasi:

- **Reguler:** [Hari], [Jam] di [Tempat]
- **Online:** [Hari], [Jam] via [Zoom/Meet]
- **Emergency:** Via WA (response within 24 hours)

Communication Channels:

- **LMS:** [Link ke LMS]
 - **WhatsApp Group:** [Link]
 - **GitHub Organization:** [Link jika ada]
 - **Email:** Use for formal submission
-

Q. ACADEMIC INTEGRITY

Plagiarism Policy:

ZERO TOLERANCE untuk plagiarism. Konsekuensi:

- Copy dari kelompok lain: Nilai E untuk kedua kelompok
- Copy dari internet tanpa citation: Nilai E
- Similarity > 30% (Turnitin): Review dan kemungkinan nilai E

Allowed:

- Diskusi konsep dengan kelompok lain

- Gunakan library/framework existing (dengan citation)
- Referensi ke textbook/paper (dengan citation)
- Stack Overflow untuk debugging (acknowledge in code)

NOT Allowed:

- Copy code dari kelompok lain
- Copy-paste code dari GitHub/internet tanpa pemahaman
- Beli/download solution dari internet
- Satu anggota mengerjakan semua, yang lain tidak kontribusi

Citation Guidelines:

- Cite semua sources (books, papers, websites, code repositories)
- Use proper citation format (APA/IEEE)
- In-code comments untuk acknowledge borrowed code snippets
- Include all references in bibliography

R. EVALUATION RUBRIC

Rubric Detail per Komponen:

Teori Dasar (20 points)

| Kriteria | Excellent (18-20) | Good (15-17) | Satisfactory (12-14) | Poor (<12) |
|---------------|---------------------------------|-----------------------|-----------------------|--------------------|
| Completeness | All concepts covered in depth | Most concepts covered | Some concepts missing | Many gaps |
| Understanding | Deep understanding, can explain | Good understanding | Surface understanding | Memorization only |
| Examples | Multiple clear examples | 2-3 examples | 1-2 examples | No examples |
| Documentation | Excellent mind map + notes | Good documentation | Basic documentation | Poor documentation |

Implementasi (25 points)

| Kriteria | Excellent (23-25) | Good (19-22) | Satisfactory (15-18) | Poor (<15) |
|-------------|----------------------------|----------------|----------------------|-------------|
| Correctness | Perfect, handles all cases | Mostly correct | Some bugs | Many errors |

| Kriteria | Excellent (23-25) | Good (19-22) | Satisfactory (15-18) | Poor (<15) |
|---------------|----------------------------------|------------------|----------------------|-------------|
| Code Quality | Clean, modular, well-commented | Good structure | Acceptable | Messy |
| Efficiency | Optimal algorithm implementation | Good performance | Acceptable | Inefficient |
| Documentation | Complete docs + tests | Good docs | Basic docs | No docs |

Analisis (15 points)

| Kriteria | Excellent (14-15) | Good (12-13) | Satisfactory (9-11) | Poor (<9) |
|----------------|------------------------------|---------------------|----------------------|-------------------|
| Depth | Insightful analysis | Good analysis | Surface analysis | No analysis |
| Visualization | Excellent plots, informative | Good visuals | Basic plots | Poor/no visuals |
| Interpretation | Clear insights | Good interpretation | Basic interpretation | No interpretation |

Presentasi (20 points)

| Kriteria | Excellent (18-20) | Good (15-17) | Satisfactory (12-14) | Poor (<12) |
|-----------------|-------------------------|--------------------|----------------------|---------------|
| Clarity | Crystal clear, engaging | Clear presentation | Understandable | Confusing |
| Demo | Flawless demo | Smooth demo | Some issues | Demo failed |
| Q&A | Excellent answers | Good answers | Acceptable | Cannot answer |
| Time Management | Perfect timing | Good timing | Slightly off | Way off |

S. SUCCESS STORIES & EXAMPLES

Contoh Project Excellent dari Tahun Sebelumnya:

1. Optimasi Rute Delivery E-Commerce (Dijkstra)

- Dataset: 500 delivery points Jakarta
- Hasil: 30% reduction in delivery time
- Bonus: Interactive web dashboard

2. Production Planning UMKM (Simpleks)

- Dataset: 3 products, 5 constraints
- Hasil: 25% increase in profit

- Bonus: Sensitivity analysis

3. Nurse Scheduling (Branch & Bound)

- Dataset: 20 nurses, 7 days
 - Hasil: Optimal schedule found
 - Bonus: Comparison with heuristics
-

T. FINAL CHECKLIST

Sebelum submit final deliverables, pastikan:

Laporan:

- Semua bab lengkap (I-VI)
- Min 20 halaman, max 30 halaman
- Semua gambar ada caption dan dijelaskan
- Referensi min 10 sumber (APA/IEEE format)
- Turnitin similarity < 20%
- Proofread, no typos

Code:

- GitHub repo public/accessible
- README.md lengkap
- Requirements.txt included
- Code runs without errors
- All functions documented
- Tests included
- Sample dataset included

Presentasi:

- Slide 15-20 pages
- Clear structure (intro-theory-demo-results-conclusion)
- All members have speaking parts
- Demo tested and works
- Backup plan ready (video/screenshots)
- Time rehearsed (15 minutes ±1 min)

Deliverables:

- All files named correctly
 - Uploaded to correct LMS location
 - All links working (GitHub, video, Drive)
 - Confirmation email sent
-

Catatan: Pastikan setiap anggota kelompok berkontribusi secara aktif dan merata dalam penyelesaian tugas ini.

Selamat bekerja dan semoga sukses! 🎓

Last Updated: [Tanggal]

Document Version: 1.0

Course: Riset Operasi - Teknik Informatika