# **Pre-Coding Essentials (Component: crates/vm\_algo/src/tabulation/ranked\_condorcet.rs, Version/FormulaID: VM-ENGINE v0) — 41/89**

## **1) Goal & Success**

Goal: Deterministically tabulate **Condorcet** per unit: build full **pairwise** matrix from ranked ballots; if a Condorcet winner exists, pick it; otherwise apply the configured **completion rule** (schulze or minimax).

Success: Integer-only counts, canonical option order, audit pairwise matrix; no RNG (cycles resolved by the completion rule, not by tie policy).

## **2) Scope**

In scope: Per-unit pairwise tallying; winner detection; completion rule executor (Schulze or Minimax); deterministic secondary ordering where completion rule needs tie-breaks; emit audit structures.

Out of scope: Allocation, gates math, I/O/schema parsing.

## **3) Inputs → Outputs**

Inputs:

ballots: &[(Vec<OptionId>, u64)] (validated ranked groups)

options: &[OptionItem] (canonical (order\_index, id) order)

turnout: Turnout

params: &Params (reads VM-VAR-001=ranked\_condorcet, VM-VAR-005 completion)

Outputs:

(UnitScores, Pairwise, CondorcetLog) where UnitScores.scores holds the winner-only final tallies (or final round tallies per rule), Pairwise is the audit matrix.

## **4) Entities/Tables (minimal)**

## **5) Variables (used here)**

## **6) Functions (signatures only)**

rust

CopyEdit

use std::collections::BTreeMap;

use vm\_core::{

ids::{UnitId, OptionId},

entities::{Turnout, OptionItem},

variables::Params,

};

pub struct Pairwise { pub wins: BTreeMap<(OptionId, OptionId), u64> }

pub struct CondorcetLog {

pub completion\_rule: CompletionRule, // Schulze or Minimax

pub winner: OptionId,

pub pairwise\_summary: Pairwise,

}

#[derive(Copy, Clone, Eq, PartialEq)]

pub enum CompletionRule { Schulze, Minimax }

pub fn tabulate\_ranked\_condorcet(

unit\_id: UnitId,

ballots: &[(Vec<OptionId>, u64)],

options: &[OptionItem],

turnout: Turnout,

params: &Params,

) -> (UnitScores, Pairwise, CondorcetLog);

// Internals

fn build\_pairwise(ballots: &[(Vec<OptionId>, u64)], options: &[OptionItem]) -> Pairwise;

fn condorcet\_winner(pw: &Pairwise, options: &[OptionItem]) -> Option<OptionId>;

fn schulze\_winner(pw: &Pairwise, options: &[OptionItem]) -> OptionId;

fn minimax\_winner(pw: &Pairwise, options: &[OptionItem]) -> OptionId;

## **7) Algorithm Outline (implementation plan)**

**Canonical ordering**: work with options sorted by (order\_index, id); all maps are BTree\* for stable iteration.

**Pairwise tally**: for each ballot group, for each ordered pair (A,B) where A is ranked above B, add count to wins[(A,B)]; abstain when neither is ranked (no increment). Produce complete matrix for audit.

**Winner detection**: if some X has wins[(X,Y)] > wins[(Y,X)] for all Y≠X, return **Condorcet winner**.

**No Condorcet winner → completion**:

If VM-VAR-005=schulze: compute strongest paths and pick maximal per Schulze relation.

If …=minimax: pick option minimizing its **maximum** pairwise defeat.

Where internal ties arise inside the method, break **deterministically** by (order\_index, id) (RNG is not used for cycles).

**Assemble**: UnitScores with winner-only score (e.g., put winner’s tally = turnout.valid\_ballots, others 0) or a final tally representation consistent with report needs; return (UnitScores, Pairwise, CondorcetLog).

## **8) State Flow**

Feeds pipeline **TABULATE → (winner for unit)**, then **ALLOCATE/AGGREGATE** as usual; decision gates are independent and follow Doc 4 rules.

## **9) Determinism & Numeric Rules**

Integer counts only; no floats; stable data structures; completion rule is algorithmic (not tie policy).

## **10) Edge Cases & Failure Policy**

**Empty/zero-valid**: no pairwise comparisons; select smallest (order\_index,id) as degenerate outcome.

**Equal-rank / truncation**: assumed pre-validated by loader; if encountered, treat unranked comparisons as abstentions.

**All pairwise ties**: completion rule reduces to deterministic order fallback.

## **11) Test Checklist (must pass)**

**Annex B Condorcet (Schulze)**: ballot profile yields **winner B** under schulze as specified.

Switch completion to **minimax** on the same profile to verify a different (or same) winner per rule mechanics.

Determinism: permute ballot order / option IDs → identical Pairwise matrix and winner.

Degenerate cases (no rankings, all ties) choose deterministic fallback without RNG.