Untitled

Quick list (copy/paste this block, then prune what you really need)

```
lean
CopyEdit
import Init.WF -- WellFounded, Acc, InvImage.wf, Subrelation.wf
import Batteries.WF -- makes WellFounded.fix / Acc.rec compile to code (optional but handy)
import Mathlib/Data/Nat/Order/Basic -- lemmas about <, ≤, succ, lt_succ_self, etc. on ℕ
import Mathlib/Data/Prod/Lex -- Prod.Lex, its constructors, and order instances
import Mathlib/SetTheory/Ordinal/Basic -- ordinals themselves
import Mathlib/SetTheory/Ordinal/Arithmetic -- ordinal +, ·, ^, etc.
import Mathlib/Tactic/Linarith/Frontend -- linarith
import Mathlib/Tactic/Ring/Basic -- ring, ring_nf, etc.
```

If you just want “generic tactics”, `import Mathlib.Tactic` is OK, but it drags in a lot. Be surgical instead.

Line-by-line “wrong import” check

What you wrote	Status	Use instead / keep	Why / Where it lives
`import` Mathlib.Data.Nat.Order	**Wrong / split**	`import Mathlib/Data/Nat/Order/Basic` (and maybe `/Lemmas`) or just `Mathlib/Data/Nat/Basic` if you only need trivial stuff	The nat order material was split: `.../Order/Basic` & `.../Order/Lemmas`. GitLabGitLab
`import` Mathlib.Data.Prod.Lex	**Correct**	Keep it	Gives `Prod.Lex` and order instances. Floris van Doorn
`import` Mathlib.Data.Ordinal.Basic	**Wrong path**	`import Mathlib/SetTheory/Ordinal/Basic`	Ordinals live under `SetTheory`. leanprover-community.github.ioleanprover-community.github.io
`import` Mathlib.Data.Ordinal.Arithmetic	**Wrong path**	`import Mathlib/SetTheory/Ordinal/Arithmetic`	Same reason. leanprover-community.github.io
`import Mathlib.Tactic`	**Technically OK but huge**	Prefer targeted imports: `Mathlib/Tactic/Linarith/Frontend`, `Mathlib/Tactic/Ring/Basic`, etc.	`linarith` doc: leanprover-community.github.io; `ring` doc: leanprover-community.github.io
`import` Init.WF	**Correct**	Keep it	This is where `WellFounded`, `Acc`, InvImage.wf, Subrelation.wf, `measure` live now. leanprover-community.github.ioGitHub
`import Mathlib.Tactic.Linarith`	**Use sub-file**	`import Mathlib/Tactic/Linarith/Frontend` (or `.../Preprocessing`, etc., if you need internals)	The main tactic entry point is in `Frontend`. leanprover-community.github.iomath.iisc.ac.in
`import Mathlib.Tactic.Ring`	**Use sub-file**	`import Mathlib/Tactic/Ring/Basic` (and optionally `/RingNF`)	Mathlib4 split ring tactic into submodules. leanprover-community.github.ioleanprover-community.github.io

`import` Std.Data.WellFounded	**Gone**	Use Init.WF + Batteries.WF	Std was retired; Batteries hosts the compiled versions. leanprover-community.github.ioleanprover-community.github.io
`measure_wf`, `WellFounded.invImage`, `WellFounded.subrel`, `Prod.lex_wf`	**All renamed / moved**	Use InvImage.wf, Subrelation.wf, and instances from `Prod.instWellFoundedRelation`	See Lean core docs for InvImage.wf; Nat.wfRel: leanprover-community.github.ioGitHub. Batteries note: leanprover-community.github.io. There is no `Prod.lex_wf` constant—build WF via instances. Floris van Doorn

Why your step proof goals wouldn't unify (the Lex.left/right spam)

Those "type mismatch … expected Rel (rank …) got Prod.Lex …" errors are *not* solved by imports. They're a unification problem:

1. Your goal is `StepRank b a` which *defines* to `Rel (rank b) (rank a)`.
2. Lean keeps that def opaque. So `refine Prod.Lex.left ?_` can't see the pair.
3. Fix: **force the def to unfold at the goal line**:

```
theorem step_rank {a b : Trace} (h : Step a b) : StepRank b a := by
change Rel (rank b) (rank a) -- 🔴 essential
cases h with
| R_int_delta t =>
-- choose the correct constructor depth
refine Prod.Lex.left ?_ -- outer pairLex
refine Prod.Lex.left ?_ -- inner pairLex
simp [rank, eqCnt, δCnt, bulk] ; exact Nat.lt_succ_self _
-- …and so on
```

Do that once per case and 90% of your "invalid argument name 'r'" / "type mismatch Lex.left" disappears. The rest are trivial Nat inequalities—one `simp` + one `Nat` lemma (e.g. `Nat.lt_succ_self _`, `Nat.add_lt_add_left`) each. Stop shoving 10 things into `[simp …]`; that's why you get the unused-arg linter.

Constants that changed name (double-check your code)

Old name you used	New / existing thing
`measure_wf`	InvImage.wf `f Nat.lt_` wfRel.wf (or write `WellFounded.measure` yourself) leanprover-community.github.ioGitHub
`WellFounded.invImage` (as a field)	InvImage.wf (a def in Init.WF) leanprover-community.github.io
`WellFounded.subrel`	Subrelation.wf (same file) leanprover-community.github.io
`Prod.lex_wf`	no such constant; use `(Prod.instWellFoundedRelation _ _).wf` (or build with InvImage.wf) Floris van Doorn

If you *do* this with ordinals

If you pivot to the ordinal interpretation: the only two imports you really need are

```
lean
CopyEdit
import Mathlib/SetTheory/Ordinal/Basic
import Mathlib/SetTheory/Ordinal/Arithmetic
```

Everything ordinal-related hangs under `Mathlib/SetTheory/Ordinal`. leanprover-community.github.ioleanprover-community.github.io

Batteries note

You keep seeing slow `WellFounded.fix` codegen? Pull in:

```
import Batteries.WF
```

This file “exports no public defs” but flips compiler switches so `Acc.rec` / `WellFounded.fix` produce compiled code instead of massive bytecode blobs. leanprover-community.github.io