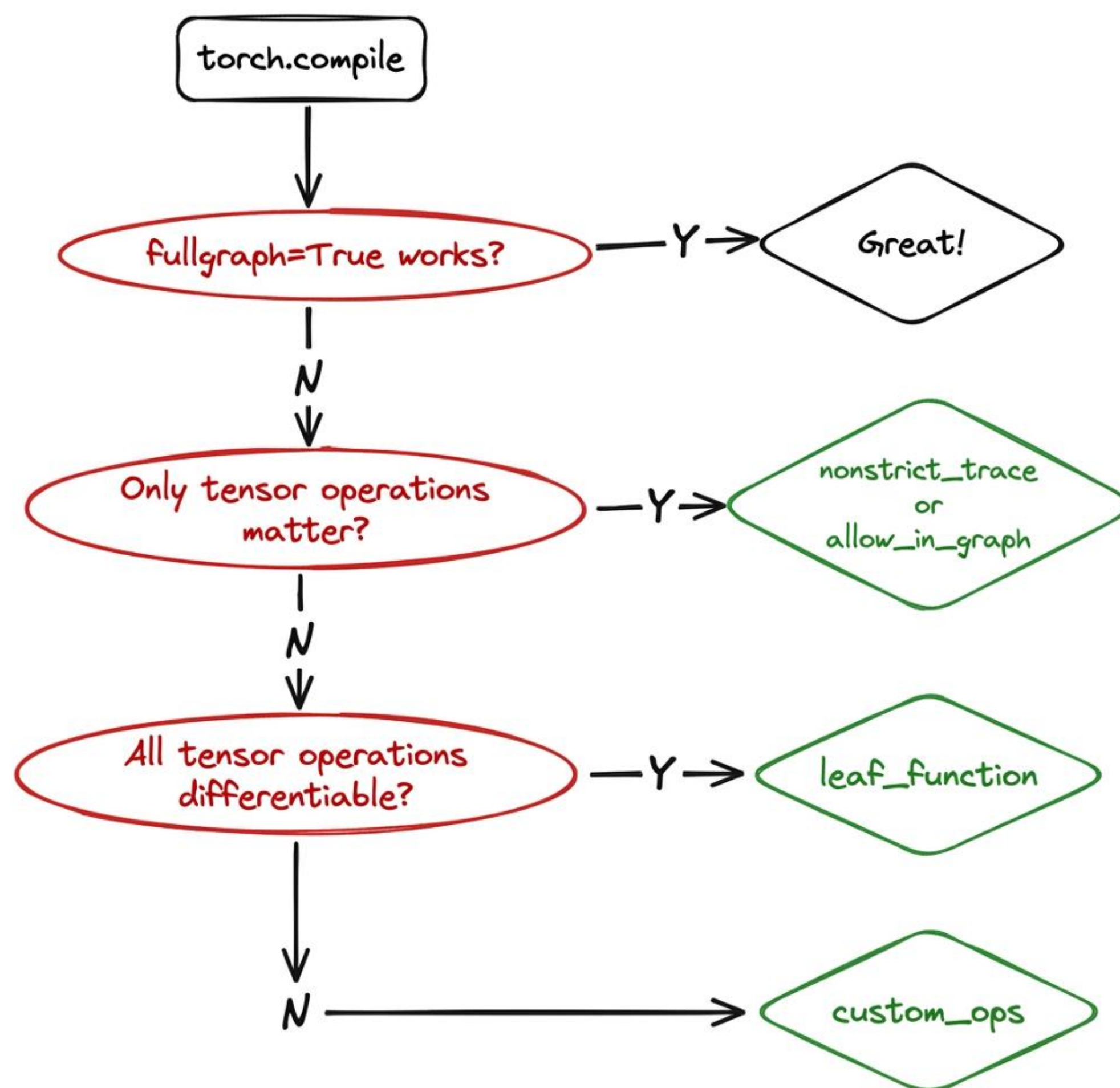


Encountering rough edges?

- **Dynamo**
 - CPython gaps, copy.deepcopy, enum construction
 - C++ libraries
 - ...
- **AOTAutograd**
 - Data-dependent control flows
 - `untyped_storage().resize()`
 - ...
- **Logging, printing in python not preserved?**
 - Non-tensor ops, e.g. logging; side effects are not preserved

Escape hatches to the rescue!



fullgraph=False

Behavior: Falling back to eager for code that dynamo doesn't support.

Use cases: We don't want to spend ANY time on improving performance.

Pros: No user efforts.

Cons: Performance can be sub-optimal.

allow_in_graph / nonstrict_trace

Behavior: Both bypass dynamo. Operators inside are compiled.

``allow_in_graph`` registers the original function globally;

``nonstrict_trace`` registers the wrapper, with pytree support.

Use cases: Operations that Dynamo doesn't know how to handle e.g. enum construction, copy.deepcopy.

Pros: best performance among the other escape hatches.

Cons: python side effects are gone. May not always be safe.

```
@torch._dynamo.nonstrict_trace
def trace_me(self, t):
    torch._dynamo.graph_break()
    return t + self
```

leaf_function (new!)

Behavior: Don't compile. Call the original callable as an opaque Python object in the graph.

Use cases: preserve side-effectful Python code in the graph, e.g. printing/logging

Pros: preserve the python behavior

Cons: no compilation.

```
@leaf_function
def nonzero_forward(mod, x):
    out = mod.linear(x)
    nonzero_indices = (out > 0).nonzero()
    return (out, nonzero_indices)

@nonzero_forward.register_fake
def nonzero_forward_fake(mod, x):
    out = mod.linear(x)
    return out, (out > 0).nonzero()
```

	allow_in_graph	nonstrict_trace	leaf_function (new!)	custom_ops
Dynamo	Opaque			
AOTAutograd	Trace into		Opaque	
Inductor	Compile		Not supported yet	Opaque
Export	Trace into		Opaque	
Input types	Fx-proxyable types	Pytree of fx-proxyable types, nn.Module		Fx-proxyable types
Output types	Fx-proxyable types	Pytree of fx-proxyable types		Fx-proxyable types
Side effects (logging, IO)	Side effects will be ignored at runtime		Side effects are preserved	

Opaque: the annotated function is a black box. **Trace into:** the annotated function is traced into by AOTAutograd; the operators in the function are captured in the AOT graph. **Compile:** after AOTAutograd, Inductor sees the AOT graph for the operators inside the function, so they are compiled by Inductor. **Pytree:** nested lists / dataclasses / dicts of fx-proxyable types.