```
from dataclasses import dataclass, field
from typing import Any, Dict, List, Tuple, Callable

@dataclass
class Frame:
    tensor: Any
    timestamp_us: int
    exposure_time_us: int
    iso: int
    camera_id: str = "wide"
    pose: Any = None
    intrinsics: Any = None
    ois_gyro: Any = None
    metadata: Dict[str, Any] = field(default_factory=dict)

@dataclass
class Flow:
    frames: List[Frame]
    alignment: Dict[str, Any] = field(default_factory=dict)

class Node:
    def __init__(self, node_id: str, fn: Callable, params: Dict[str, Any] = None):
        self.id = node_id
        self.fn = fn
        self.params = params or {}

    def __call__(self, x):
        return self.fn(x, **self.params)

class Graph:
    def __init__(self):
        self.nodes: Dict[str, Node] = {}
        self.edges: List[Tuple[str, str]] = []

    def add_node(self, node: Node):
        self.nodes[node.id] = node

    def add_edge(self, u: str, v: str):
        self.edges.append((u, v))

    def run(self, inputs: Dict[str, Any]):
        state = dict(inputs)
        order = self.topo_sort()
        for nid in order:
```

```
            node = self.nodes[nid]
            x = state.get(nid, state.get('input'))
            out = node(x)
            state[nid] = out
        return {nid: state[nid] for nid in order}

    def topo_sort(self) -> List[str]:
        from collections import defaultdict, deque
        indeg = defaultdict(int)
        adj = defaultdict(list)
        for u, v in self.edges:
            indeg[v] += 1
            adj[u].append(v)
            if u not in indeg:
                indeg[u] = indeg[u]
        q = deque([n for n in indeg if indeg[n] == 0])
        order = []
        while q:
            u = q.popleft()
            order.append(u)
            for v in adj[u]:
                indeg[v] -= 1
                if indeg[v] == 0:
                    q.append(v)
        return order

# Example ops
def dpc_op(frame: Frame, method: str = "classic"):
    # TODO: implement dead pixel correction
    return frame

def hdr_fusion_op(flow: Flow, method: str = "content_adaptive", highlight_protect: bool = True):
    # TODO: implement HDR fusion
    return Frame(tensor=None, timestamp_us=0, exposure_time_us=0, iso=0)

def build_demo():
    g = Graph()
    g.add_node(Node("dpc", dpc_op, {"method": "classic"}))
    g.add_node(Node("hdr_fusion", hdr_fusion_op, {}))
    g.add_edge("dpc", "hdr_fusion")
    return g
```