tf.case

```
case(
    pred_fn_pairs,
    default=None,
    exclusive=False,
    strict=False,
    name='case'
)
```

Defined in `tensorflow/python/ops/control_flow_ops.py`.

See the guide: Control Flow > Control Flow Operations

Create a case operation.

The `pred_fn_pairs` parameter is a dict or list of pairs of size N. Each pair contains a boolean scalar tensor and a python callable that creates the tensors to be returned if the boolean evaluates to True. `default` is a callable generating a list of tensors. All the callables in `pred_fn_pairs` as well as `default` (if provided) should return the same number and types of tensors.

If `exclusive==True`, all predicates are evaluated, and an exception is thrown if more than one of the predicates evaluates to `True`. If `exclusive==False`, execution stops at the first predicate which evaluates to True, and the tensors generated by the corresponding function are returned immediately. If none of the predicates evaluate to True, this operation returns the tensors generated by `default`.

`tf.case` supports nested structures as implemented in `tensorflow.python.util.nest`. All of the callables must return the same (possibly nested) value structure of lists, tuples, and/or named tuples. Singleton lists and tuples form the only exceptions to this: when returned by a callable, they are implicitly unpacked to single values. This behavior is disabled by passing `strict=True`.

If an unordered dictionary is used for `pred_fn_pairs`, the order of the conditional tests is not guaranteed. However, the order is guaranteed to be deterministic, so that variables created in conditional branches are created in fixed order across runs.

Example 1:

Pseudocode:

```
if (x < y) return 17;
else return 23;
```

Expressions:

```
f1 = lambda: tf.constant(17)
f2 = lambda: tf.constant(23)
r = case([(tf.less(x, y), f1)], default=f2)
```

Example 2:

Pseudocode:

```
if (x < y && x > z) raise OpError("Only one predicate may evaluate true");
if (x < y) return 17;
else if (x > z) return 23;
else return -1;
```

Expressions:

```
def f1(): return tf.constant(17)
def f2(): return tf.constant(23)
def f3(): return tf.constant(-1)
r = case({tf.less(x, y): f1, tf.greater(x, z): f2},
         default=f3, exclusive=True)
```

Args:

- `pred_fn_pairs` : Dict or list of pairs of a boolean scalar tensor and a callable which returns a list of tensors.
- `default` : Optional callable that returns a list of tensors.
- `exclusive` : True iff at most one predicate is allowed to evaluate to `True` .
- `strict` : A boolean that enables/disables 'strict' mode; see above.
- `name` : A name for this operation (optional).

Returns:

The tensors returned by the first pair whose predicate evaluated to True, or those returned by `default` if none does.

Raises:

- `TypeError` : If `pred_fn_pairs` is not a list/dictionary.
- `TypeError` : If `pred_fn_pairs` is a list but does not contain 2-tuples.
- `TypeError` : If `fns[i]` is not callable for any i, or `default` is not callable.
- `ValueError` : If in eager mode and all predicates are false and no default is provided.
- `ValueError` : If in eager mode and is passed a dictionary.