A content-addressed, append-only store that supports verifiable, linked histories of updates.

Caveats: linear history, uniqueness of stored values, everything is unambiguously serialisable and that serialisation format has a deterministic diffing ability.

(1) Adding a new value

```
let v1 = {flights:[…], tx: null} in
let h1, vhash1 = Store.add store v1 in …
```

Previous Store

H(diff) !	None
H(v1)	H(diff)

Diff Store

H(diff)	{ vhash: H(diff), diff: diff({}, v1) }

(!) `H(diff)` is some root hash we store, calculated as the hash of the initial diff. It just has to be unique.

(2) Adding a new version of a value

```
let v2 = {flights:[…], tx: Some “ABCD”} in
let h2, vhash2 = Store.add ~prev:h1 store v2 in …
```

Previous Store

H(diff)	None
H(v1)	H(diff)
H(v2)	H(v1)

Diff Store

H(diff)	{vhash: H(diff), diff: diff({}, v1)}
H(v1)	{vhash: H(v1+prev_vhash), diff: diff(v1, v2)}

(3) Get latest value of older version

```
let h_latest, vhash_latest = Store.latest store h1 in …
```

Previous Store

H(diff) 3	None
H(v1) 1	2 H(diff)
H(v2)	H(v1)

Diff Store

4 H(diff)	{vhash: H(diff), diff: diff({}, v1)}
5 H(v1)	{vhash: H(v1+prev_vhash), diff: diff(v1, v2)}

1. The user has asked for the latest value of the value stored at `h1 = H(v1)`
2. We find the previous link and move backwards.
3. We do this until we’re at the root i.e. looking up previous returns None
4. We now lookup the root in the diff store.
5. We move through the diff store by folding over the entries, crucially this works because `H(apply(acc, next_diff)) = H(vN)` where *acc* is the accumulated value after applying diffs. We do this until looking up the next diff is not found in the diff store.

(4) Verify the latest value is indeed the latest value of our initial value

```
let history = Store.history store h1 in
Let verified = verify ~start:(h1, vhash1) ~end:(h_latest, vhash_latest) history …
```

History

diff1 = diff({}, v1)
diff2 = diff(v1, v2)

`check_v1 = apply({}, diff1), check_vhash1 = H(diff1)`

`check_v2 = apply(check_v1, diff2), check_vhash2 = H(check_v1+check_vhash1)`

At some point `H(check_vn) = h1` (or whatever the start content-addressed hash is) and we check that the verified hashes line up i.e. `vhash1 = check_vhash1`. Then we keep going forward until the latest (in this case `check_v2`) and check that we’ve computed the same verifiable hash as we got.