Gas Optimization with Solady

solady.org

𝕏: optimizoor (vectorized.eth)

Github

solady.org

solady Public

Unpin | Unwatch 28 | Fork 208 | Star 1.6k

main | 1 branch | 115 tags | Go to file | Add file | Code

Vectorized Tidy FixedPointMathLib (#608) — cbcde23 1 hour ago — 715 commits

.github	Make woke ci optional (#567)	2 weeks ago
audits	Shung ERC721 audit (#498)	2 months ago
ext/woke	Ackee Blockchain & RockawayX audit tests (#456)	3 months ago
js	Add .d.ts (#604)	3 days ago
lib	Tidy (#424)	4 months ago
src	Tidy FixedPointMathLib (#608)	1 hour ago
test	ERC20 fixes and optimizations (#538)	12 hours ago
.gas-snapshot	Tidy FixedPointMathLib (#608)	1 hour ago
.gitignore	Ackee Blockchain & RockawayX audit tests (#456)	3 months ago
.gitmodules	Tidy (#424)	4 months ago
LICENSE.txt	Add first code and tests	last year
README.md	Fix solmate links (#587)	last week
foundry.toml	Update ci (#508)	2 months ago
logo.svg	Update logo.svg	10 months ago
package-lock.json	Change linter from prettier to forge fmt (#222)	10 months ago
package.json	Bump version to 0.0.122	5 hours ago

README.md

npm v0.0.122 | build passing | solidity >=0.8.4 <=0.8.21

Gas optimized Solidity snippets.

I'm soooooooOoooooooooOoooOoooooooooooooooo...

About

Optimized Solidity snippets.

- Readme
- MIT license
- Activity
- 1.6k stars
- 28 watching
- 208 forks

Releases

115 tags

Create a new release

Packages

No packages published

Publish your first package

Used by 196

+ 188

Contributors 40

+ 29 contributors

Languages

Solidity 93.6% · Python 5.5% · JavaScript 0.9%

Why?

- Save gas.
- Cool features.
- Nice APIs.

zeroknots.eth @zeroknotsETH

Went down the Solady codebase this morning. Some incredible gems in the Util Libs.

sorting an array in place? Wild!
great work @optimizoor!

Incredible, that Vectorized's low level yul code makes #solidity feel like a high level language.

Paul Razvan Berg @PaulRBerg · Jan 16

Ok so I just spent only a few minutes skimming through the **Solady** codebase, but I can confidently say that this is the most impressive Solidity repository I have ever seen.

Shout out to @optimizoor and contributors. This is great work.

Vectorized/**solady**

Optimized Solidity snippets.

40 Contributors | 197 Used by | 3 Discussions | 2k Stars | 208 Forks

github.com
GitHub - Vectorized/solady: Optimized Solidity snippets.
Optimized Solidity snippets. Contribute to Vectorized/solady development by creating an account on GitHub.

7 | 18 | 170 | 22K

Philosophy

“... what is the goal? You're trying to get people to get things done quickly. And so you need libraries, you need high quality libraries, and then you need a user base around them that can assemble them and do cool things with them. Right. And so to me, the question is, what enables high quality libraries?”

– Chris Lattner

“... These assumptions lead to the conclusion that compiler optimization advances double computing power every 18 years. QED.

This means that while hardware computing horsepower increases at roughly 60% / year, compiler optimizations contribute only 4%. Basically, compiler optimization work makes only marginal contributions.

Perhaps this means Programming Language Research should be concentrating on something other than optimizations. Perhaps programmer productivity is a more fruitful arena.”

– Todd A. Proebsting (Proebsting’s Law)

Highlights

Solidity

```
fallback() external payable {
    LibZip.cdFallback();
}

receive() external payable {
    LibZip.cdFallback();
}
```

JS / TS

```
import { LibZip } from "solady";

let ABI = [
  "function transfer(address to, uint amount)"
];

let iface = new ethers.utils.Interface(ABI);

let data = iface.encodeFunctionData("transfer", [
  "0x1234567890123456789012345678901234567890",
  parseEther("1.0")
]);

let callResult = await provider.call({
  to: "<your_contract_address>",
  data: LibZip.cdCompress(data)
});
```

Plug-and-play Calldata Compression for L2 via LibZip

Highlights

```
function getValues(address edition) public view returns (string memory) {
    Store memory s = _values[edition];
    if (s.value == address(0)) revert ValuesDoNotExist();
    bytes memory data = SSTORE2.read(s.value);
    if (s.isCompressed) data = LibZip.flzDecompress(data);
    return string(data);
}
```

LibZip + SSTORE2

Highlights

```
function testParseJWTGas() public {
    string memory jwt =
        "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwia
    string[] memory jwtSplitted = LibString.split(jwt, ".");
    JSONParserLib.Item memory header =
        JSONParserLib.parse(string(Base64.decode(jwtSplitted[0])));
    JSONParserLib.Item memory payload =
        JSONParserLib.parse(string(Base64.decode(jwtSplitted[1])));
    assertEq(jwtSplitted.length, 3);
    assertEq(jwtSplitted[2], "SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c");
    assertEq(header.at('"alg"').value(), '"HS256"');
    assertEq(header.at('"typ"').value(), '"JWT"');
    assertEq(payload.at('"sub"').value(), '"1234567890"');
    assertEq(payload.at('"name"').value(), '"John Doe"');
    assertEq(JSONParserLib.parseUint(payload.at('"iat"').value()), 1516239022);
}
```

JSONParserLib + Base64 + LibString

Highlights

```
function combine(address[] memory additions) public {
    (address[] memory current) = abi.decode(SSTORE2.read(_storageAddress), (address[]));
    LibSort.sort(current);
    LibSort.uniquifySorted(current);
    LibSort.sort(additions);
    LibSort.uniquifySorted(additions);
    address[] memory union = LibSort.union(current, additions);
    address[] memory difference = LibSort.difference(union, current);
    for (uint256 i; i != difference.length; ++i) {
        _doSomething(difference[i]);
    }
    _storageAddress = SSTORE2.write(abi.encode(union));
}
```

LibSort + SSTORE2

Highlights

```
using LibMap for *;

LibMap.Uint32Map private _uint32Map;

LibMap.Uint40Map private _timestampMap;

function setUint32(uint256 i, uint32 x) public {
    _uint32Map.set(i, x);
}

function setTimestamp(uint256 i, uint40 t) public {
    _timestampMap.set(i, t);
}
```

LibMap

Highlights

- Drop-in replacements for

  ECDSA, SignatureCheckerLib, MerkleProofLib.
- ERC1967Factory.
- LibClone.
- RedBlackTreeLib.

Codebase

- Minimal inheritance.
- Minimal dependencies.
- Great learning resource.

```
> tree src/
src/
├── Milady.sol
├── auth
│   ├── Ownable.sol
│   └── OwnableRoles.sol
├── tokens
│   ├── ERC1155.sol
│   ├── ERC20.sol
│   ├── ERC2981.sol
│   ├── ERC4626.sol
│   ├── ERC6909.sol
│   ├── ERC721.sol
│   └── WETH.sol
└── utils
    ├── Base64.sol
    ├── CREATE3.sol
    ├── Clone.sol
    ├── DateTimeLib.sol
    ├── DynamicBufferLib.sol
    ├── ECDSA.sol
    ├── EIP712.sol
    ├── ERC1967Factory.sol
    ├── ERC1967FactoryConstants.sol
    ├── FixedPointMathLib.sol
    ├── JSONParserLib.sol
    ├── LibBit.sol
    ├── LibBitmap.sol
    ├── LibClone.sol
    ├── LibMap.sol
    ├── LibPRNG.sol
    ├── LibRLP.sol
    ├── LibSort.sol
    ├── LibString.sol
    ├── LibZip.sol
    ├── MerkleProofLib.sol
    ├── MetadataReaderLib.sol
    ├── MinHeapLib.sol
    ├── Multicallable.sol
    ├── RedBlackTreeLib.sol
    ├── SSTORE2.sol
    ├── SafeCastLib.sol
    ├── SafeTransferLib.sol
    └── SignatureCheckerLib.sol

3 directories, 39 files
```

Lower Runtime Gas

```
/// @dev Returns the cube root of `x`.
/// Credit to bout3fiddy and pcaversaccio under AGPLv3 license:
/// https://github.com/pcaversaccio/snekmate/blob/main/src/utils/Math.vy
function cbrt(uint256 x) internal pure returns (uint256 z) {
    /// @solidity memory-safe-assembly
    assembly {
        let r := shl(7, lt(0xffffffffffffffffffffffffffffffff, x))
        r := or(r, shl(6, lt(0xffffffffffffffff, shr(r, x))))
        r := or(r, shl(5, lt(0xffffffff, shr(r, x))))
        r := or(r, shl(4, lt(0xffff, shr(r, x))))
        r := or(r, shl(3, lt(0xff, shr(r, x))))

        z := div(shl(div(r, 3), shl(lt(0xf, shr(r, x)), 0xf)), xor(7, mod(r, 3)))

        z := div(add(add(div(x, mul(z, z)), z), z), 3)
        z := div(add(add(div(x, mul(z, z)), z), z), 3)
        z := div(add(add(div(x, mul(z, z)), z), z), 3)
        z := div(add(add(div(x, mul(z, z)), z), z), 3)
        z := div(add(add(div(x, mul(z, z)), z), z), 3)
        z := div(add(add(div(x, mul(z, z)), z), z), 3)
        z := div(add(add(div(x, mul(z, z)), z), z), 3)

        z := sub(z, lt(div(x, mul(z, z)), z))
    }
}
```

FixedPointMathLib.cbrt

Lower Runtime Gas

```
/// @dev Returns the log10 of `x`.
/// Returns 0 if `x` is zero.
function log10(uint256 x) internal pure returns (uint256 r) {
    /// @solidity memory-safe-assembly
    assembly {
        if iszero(lt(x, 100000000000000000000000000000000000000)) {
            x := div(x, 100000000000000000000000000000000000000)
            r := 38
        }
        if iszero(lt(x, 100000000000000000000)) {
            x := div(x, 100000000000000000000)
            r := add(r, 20)
        }
        if iszero(lt(x, 10000000000)) {
            x := div(x, 10000000000)
            r := add(r, 10)
        }
        if iszero(lt(x, 100000)) {
            x := div(x, 100000)
            r := add(r, 5)
        }
        r := add(r, add(gt(x, 9), add(gt(x, 99), add(gt(x, 999), gt(x, 9999)))))
    }
}
```

FixedPointMathLib.log10

Lower Runtime Gas

```
/// @dev Returns whether `leaf` exists in the Merkle tree with `root`, given `proof`.
function verifyCalldata(bytes32[] calldata proof, bytes32 root, bytes32 leaf)
    internal
    pure
    returns (bool isValid)
{
    /// @solidity memory-safe-assembly
    assembly {
        if proof.length {
            // Left shift by 5 is equivalent to multiplying by 0x20.
            let end := add(proof.offset, shl(5, proof.length))
            // Initialize `offset` to the offset of `proof` in the calldata.
            let offset := proof.offset
            // Iterate over proof elements to compute root hash.
            for {} 1 {} {
                // Slot of `leaf` in scratch space.
                // If the condition is true: 0x20, otherwise: 0x00.
                let scratch := shl(5, gt(leaf, calldataload(offset)))
                // Store elements to hash contiguously in scratch space.
                // Scratch space is 64 bytes (0x00 - 0x3f) and both elements are 32 bytes.
                mstore(scratch, leaf)
                mstore(xor(scratch, 0x20), calldataload(offset))
                // Reuse `leaf` to store the hash to reduce stack operations.
                leaf := keccak256(0x00, 0x40)
                offset := add(offset, 0x20)
                if iszero(lt(offset, end)) { break }
            }
        }
        isValid := eq(leaf, root)
    }
}
```

MerkleProofLib.verifyCalldata

Lower Runtime Gas

```
/// @dev Sends `amount` of ERC20 `token` from `from` to `to`.
/// Reverts upon failure.
///
/// The `from` account must have at least `amount` approved for
/// the current contract to manage.
function safeTransferFrom(address token, address from, address to, uint256 amount) internal {
    /// @solidity memory-safe-assembly
    assembly {
        let m := mload(0x40) // Cache the free memory pointer.
        mstore(0x60, amount) // Store the `amount` argument.
        mstore(0x40, to) // Store the `to` argument.
        mstore(0x2c, shl(96, from)) // Store the `from` argument.
        mstore(0x0c, 0x23b872dd000000000000000000000000) // `transferFrom(address,address,uint256)`.
        // Perform the transfer, reverting upon failure.
        if iszero(
            and( // The arguments of `and` are evaluated from right to left.
                or(eq(mload(0x00), 1), iszero(returndatasize())), // Returned 1 or nothing.
                call(gas(), token, 0, 0x1c, 0x64, 0x00, 0x20)
            )
        ) {
            mstore(0x00, 0x7939f424) // `TransferFromFailed()`.
            revert(0x1c, 0x04)
        }
        mstore(0x60, 0) // Restore the zero slot to zero.
        mstore(0x40, m) // Restore the free memory pointer.
    }
}
```

SafeTransferLib.safeTransferFrom

Lower Runtime Gas

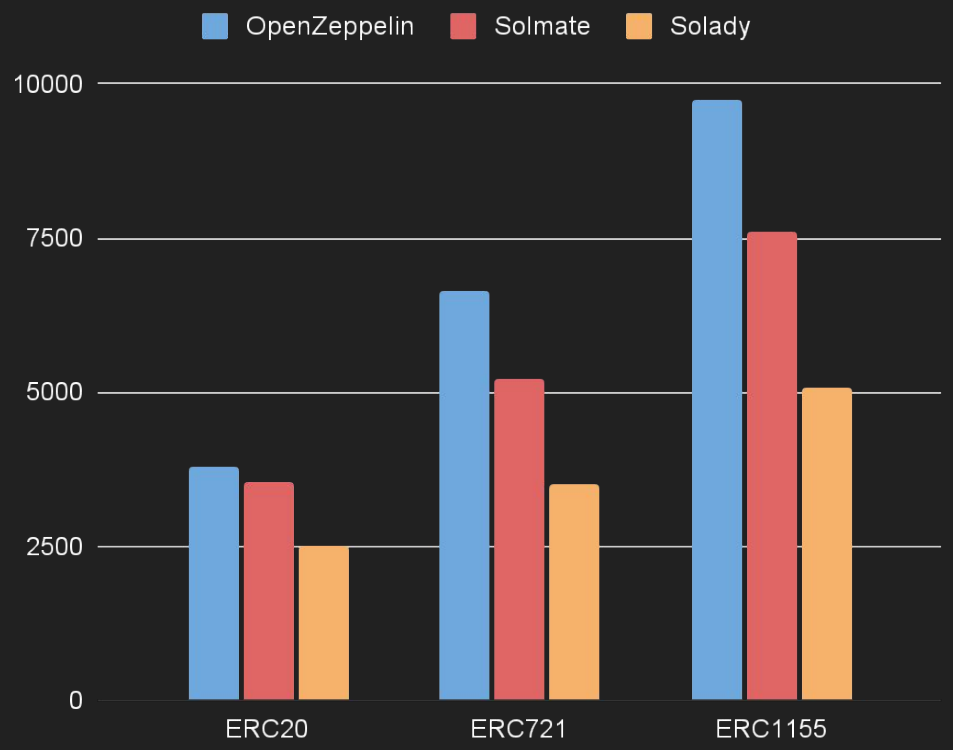
```
/// @dev Force sends `amount` (in wei) ETH to `to`, with `GAS_STIPEND_NO_GRIEF`.
function forceSafeTransferETH(address to, uint256 amount) internal {
    /// @solidity memory-safe-assembly
    assembly {
        if lt(selfbalance(), amount) {
            mstore(0x00, 0xb12d13eb) // `ETHTransferFailed()`.
            revert(0x1c, 0x04)
        }
        if iszero(call(GAS_STIPEND_NO_GRIEF, to, amount, gas(), 0x00, gas(), 0x00)) {
            mstore(0x00, to) // Store the address in scratch space.
            mstore8(0x0b, 0x73) // Opcode `PUSH20`.
            mstore8(0x20, 0xff) // Opcode `SELFDESTRUCT`.
            if iszero(create(amount, 0x0b, 0x16)) {
                returndatacopy(gas(), returndatasize(), shr(20, gas())) // For gas estimation.
            }
        }
    }
}
```

SafeTransferLib.forceSafeTransferETH

Smaller Bytecode

Pack more into a single contract.

(Spurious Dragon limit: 24576)



Barebones Contract Bytecode Size (in bytes)

ERC721

- Future-proof optimizations.
- Storage “hitchhiking”.
  - Use the extra bits in the ownership and balance slots.

emo.eth @emo_eth · Jun 20

y'all optimizing for the 2023 gas meta, @optimizoor optimizing for 2033 gas meta

vectorized.eth @optimizoor · Jun 19

Ok, this is where Solady's 721 kinda gets into crazy mode.

The custom mapping is future-proofed for Verkle tree EIP.
notes.ethereum.org/%40vbuterin/ve...

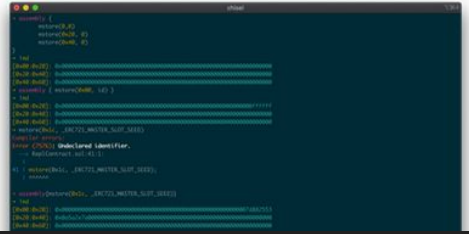
Essentially, we want each 2**32-1 consective token IDs to occupy consecutive slots to reduce the SLOAD costs in the future in case the... Show more

7858 @7858 · 2d

So the three byte ID will be in the last 3 bytes of the first word

`mstore(0x1c, _ERC721_MASTER_SLOT_SEED)` puts `0x7d8825530a5a2e7a00...` at 0x1c, which is 4 bytes shy of the end of the first word, tho

This wipes out the token ID that was just stored? v confused why the `mstore(0x00, id)` is there @emo



Etc.

- Compatible with both regular contracts and proxies (e.g. clones).

```
pragma solidity ^0.8.4;

import 'solady/src/auth/Ownable.sol';
import 'solady/src/tokens/ERC721.sol';
import 'solady/src/utils/LibString.sol';

contract TestNFT is ERC721, Ownable {
    constructor() {
        _initializeOwner(msg.sender);
    }

    function name() public view virtual override returns (string memory) {
        return "TEST NFT";
    }

    function symbol() public view virtual override returns (string memory) {
        return "TEST";
    }

    function tokenURI(uint256 id) public view virtual override returns (string memory) {
        return string(abi.encodePacked("https://remilio.org/remilio/json/", LibString.toString(id)));
    }

    function mint(address to, uint256 id) public virtual onlyOwner {
        _mint(to, id);
    }
```

Security and Correctness

- Heavily fuzz tested.
- Some math functions are formally verified.
- Audits:
  - Spearbit (Cantina) DM me on 𝕏 for the report preprint.
  - Ackee & RockawayX
  - Shung
- Make sure to read the Natspec and test with your code.

Todo

- Documentation.
- More features (6551, 4337).

Github

solady.org

solady Public

Unpin | Unwatch 28 | Fork 208 | Star 1.6k

main | 1 branch | 115 tags | Go to file | Add file | Code

Vectorized Tidy FixedPointMathLib (#608) — cbcde23 1 hour ago — 715 commits

.github	Make woke ci optional (#567)	2 weeks ago
audits	Shung ERC721 audit (#498)	2 months ago
ext/woke	Ackee Blockchain & RockawayX audit tests (#456)	3 months ago
js	Add .d.ts (#604)	3 days ago
lib	Tidy (#424)	4 months ago
src	Tidy FixedPointMathLib (#608)	1 hour ago
test	ERC20 fixes and optimizations (#538)	12 hours ago
.gas-snapshot	Tidy FixedPointMathLib (#608)	1 hour ago
.gitignore	Ackee Blockchain & RockawayX audit tests (#456)	3 months ago
.gitmodules	Tidy (#424)	4 months ago
LICENSE.txt	Add first code and tests	last year
README.md	Fix solmate links (#587)	last week
foundry.toml	Update ci (#508)	2 months ago
logo.svg	Update logo.svg	10 months ago
package-lock.json	Change linter from prettier to forge fmt (#222)	10 months ago
package.json	Bump version to 0.0.122	5 hours ago

README.md

npm v0.0.122 | build passing | solidity >=0.8.4 <=0.8.21

Gas optimized Solidity snippets.

I'm soooooooOoooooooooOoooOoooooooooooooooo...

About

Optimized Solidity snippets.

- Readme
- MIT license
- Activity
- 1.6k stars
- 28 watching
- 208 forks

Releases

115 tags

Create a new release

Packages

No packages published

Publish your first package

Used by 196

+ 188

Contributors 40

+ 29 contributors

Languages

Solidity 93.6% · Python 5.5% · JavaScript 0.9%