How To Write Fast Numerical Code

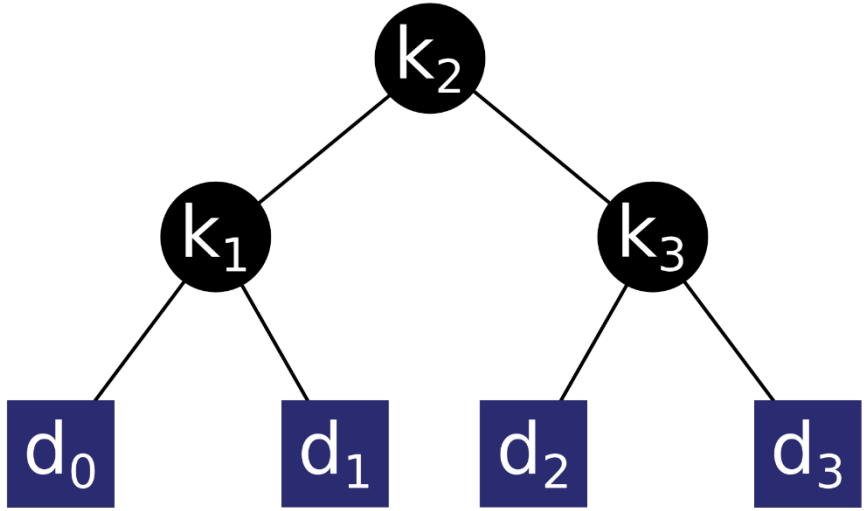
Optimal Binary Search Trees

Team 16: Jeremia Bär, Stefan Dietiker

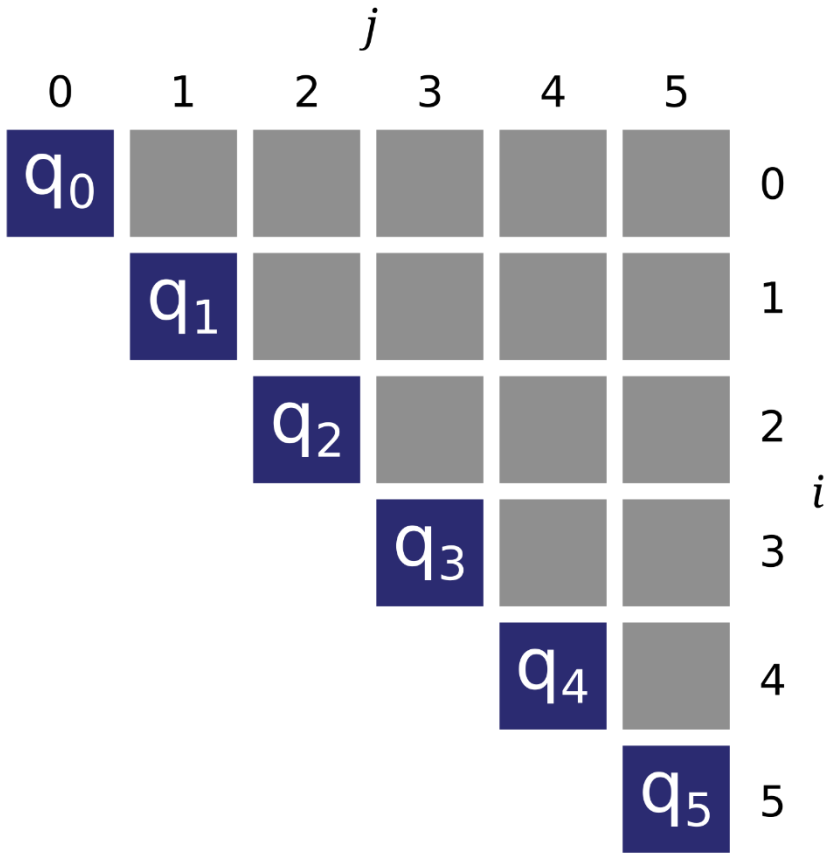
May 20th, 2014

Problem Description

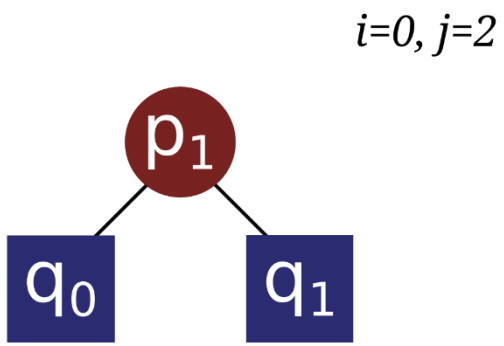
- Given:
  - Key Weights $p_1, \dots, p_n$
  - Dummy Weights $q_0, \dots, q_n$
- Required:
  - Minimize Expected Lookup Cost
  - $\sum h(k_i) \cdot p_i + \sum h(d_j) \cdot q_j$

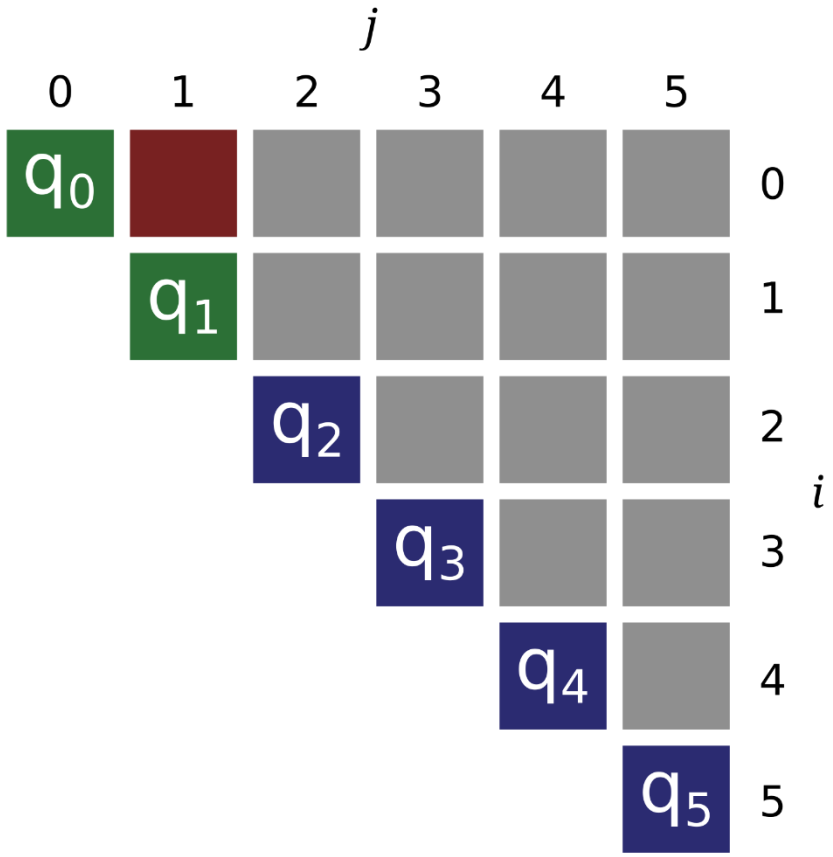


Algorithm: Dynamic Programming

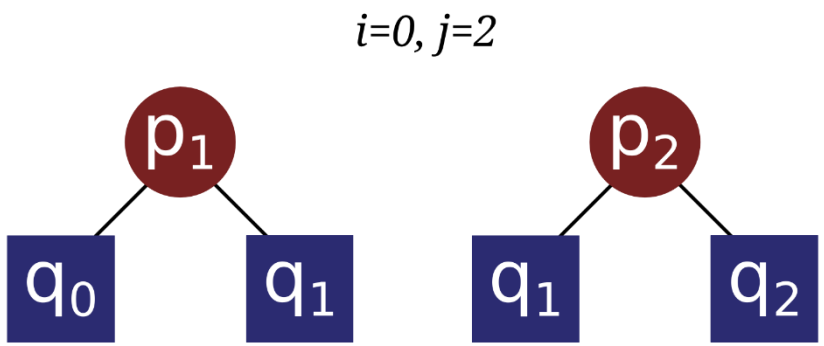


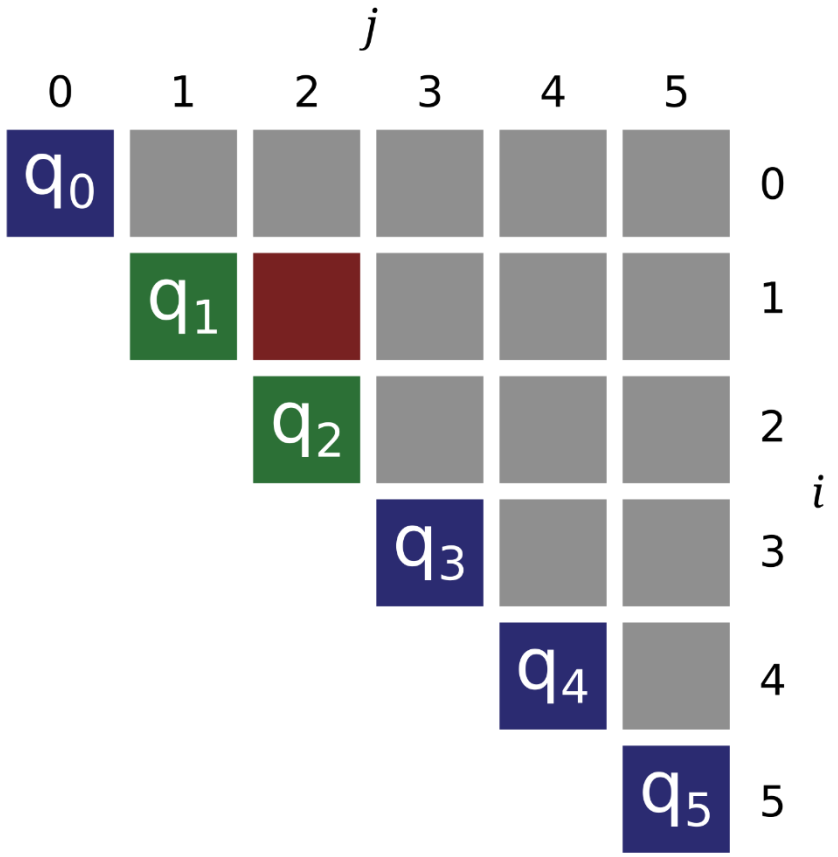
Algorithm: Dynamic Programming



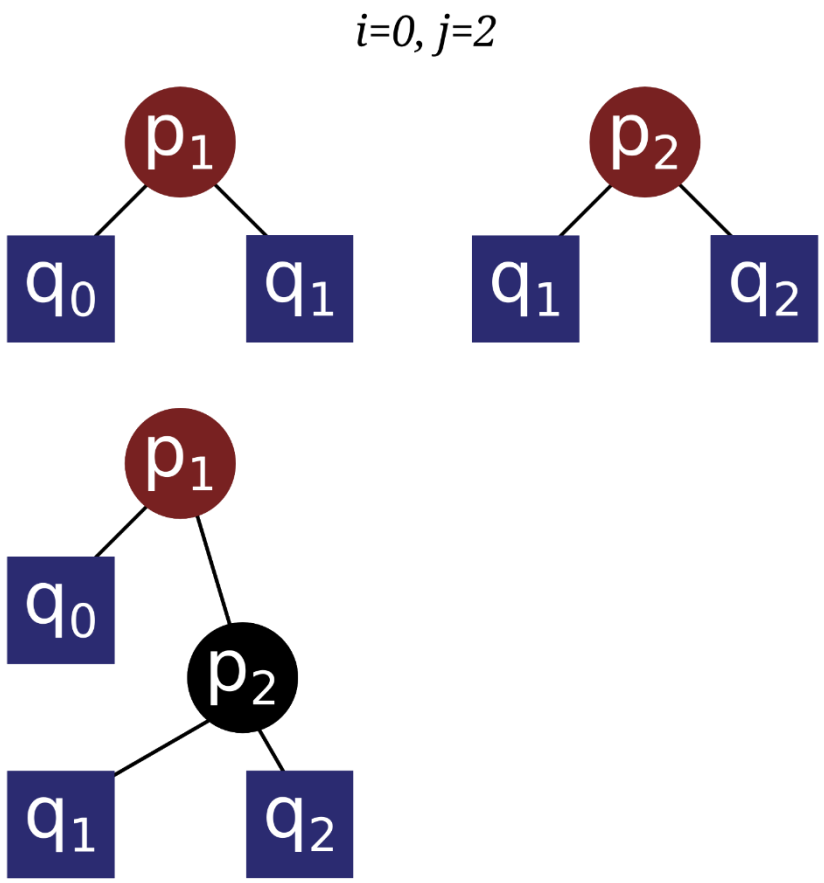


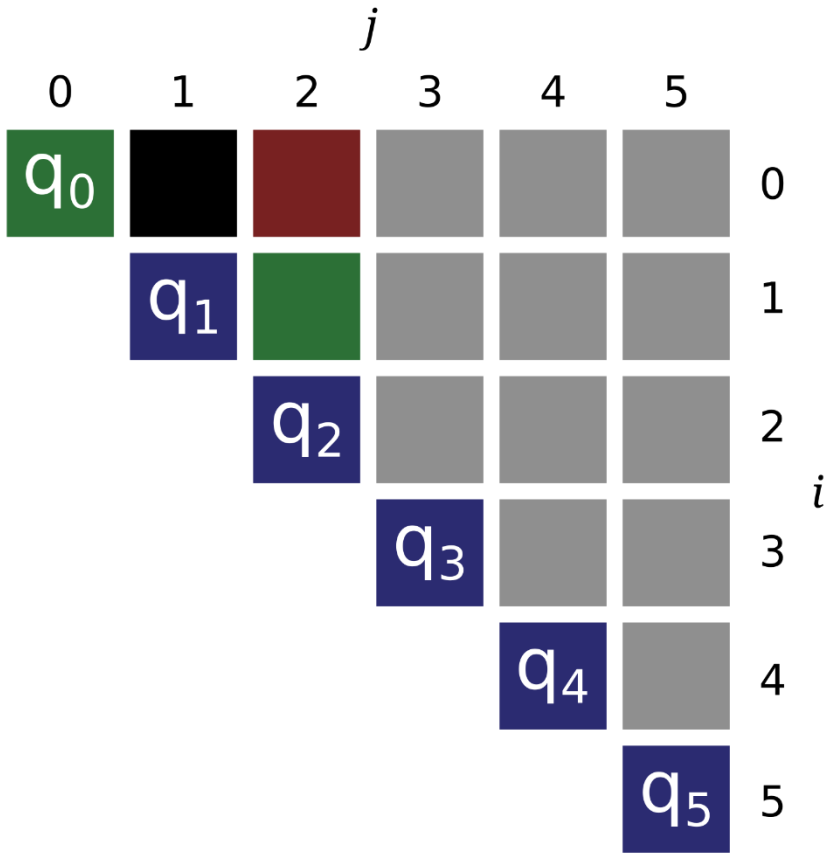
Algorithm: Dynamic Programming



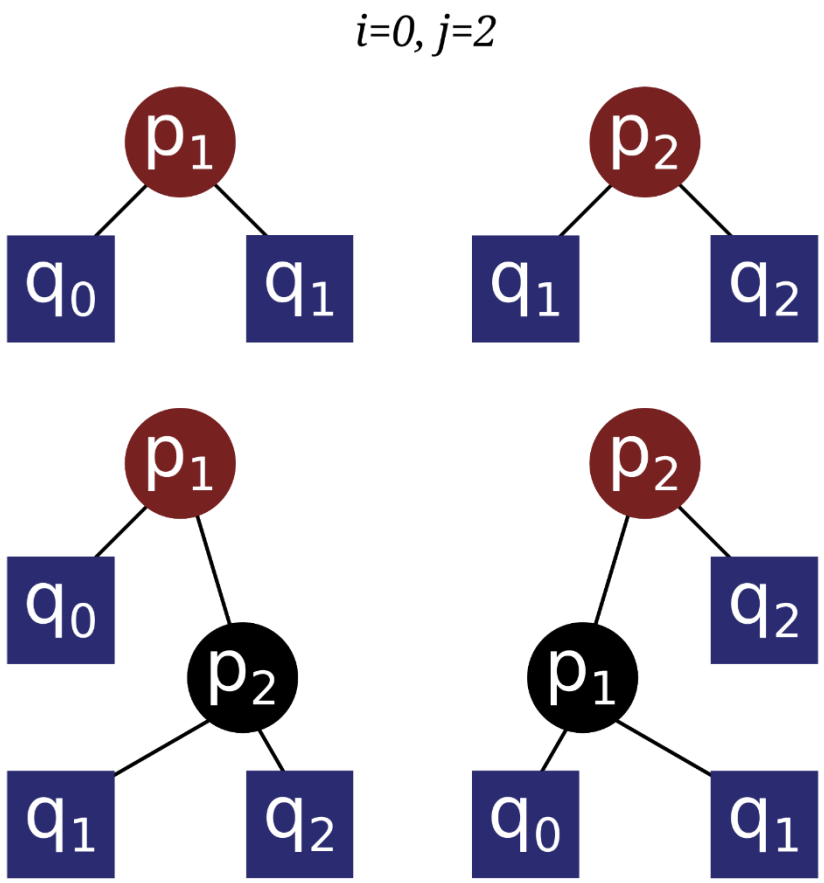


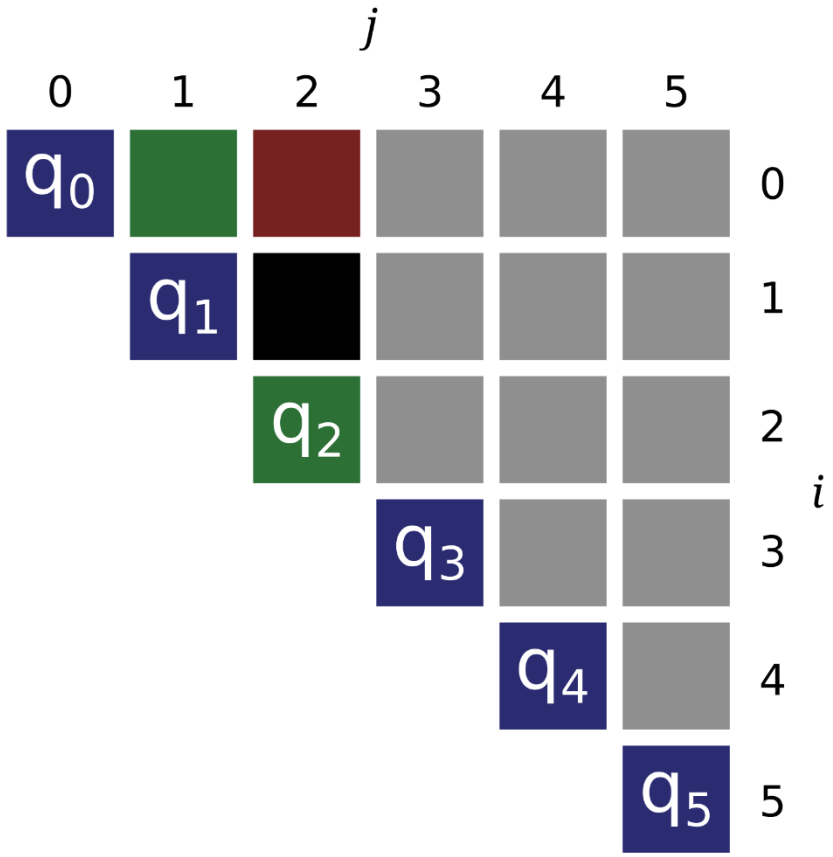
Algorithm: Dynamic Programming



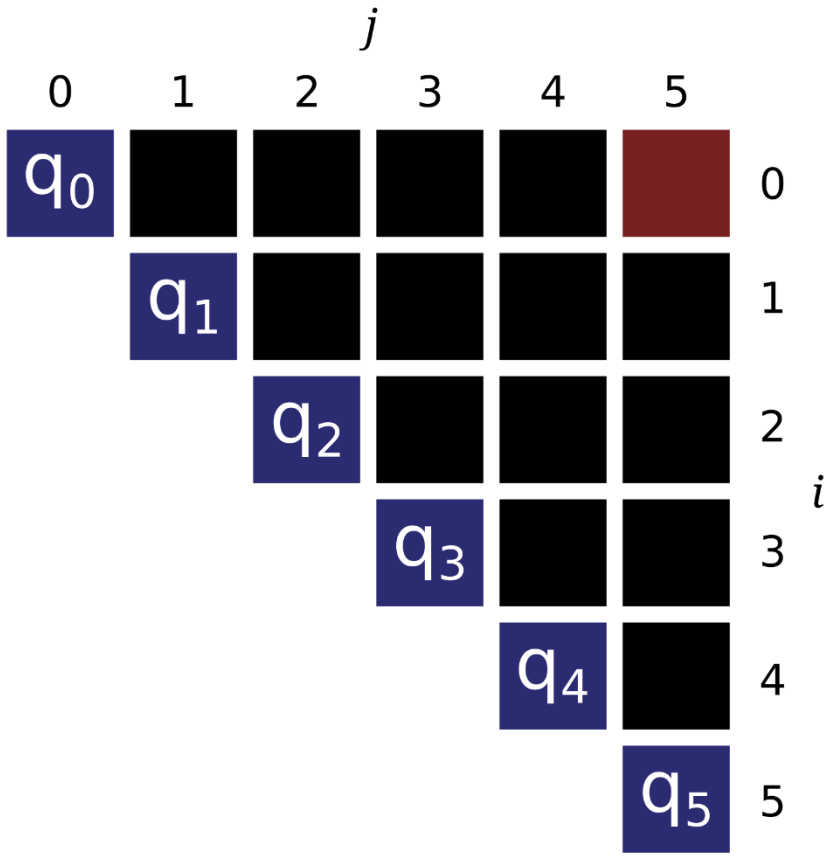


Algorithm: Dynamic Programming



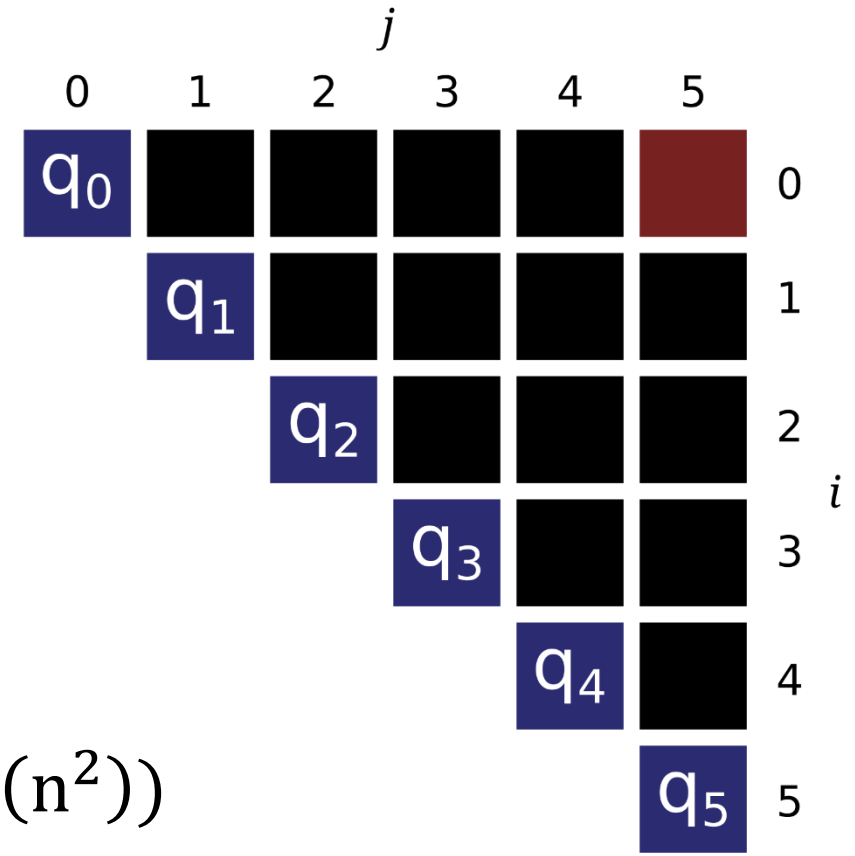


Algorithm: Dynamic Programming



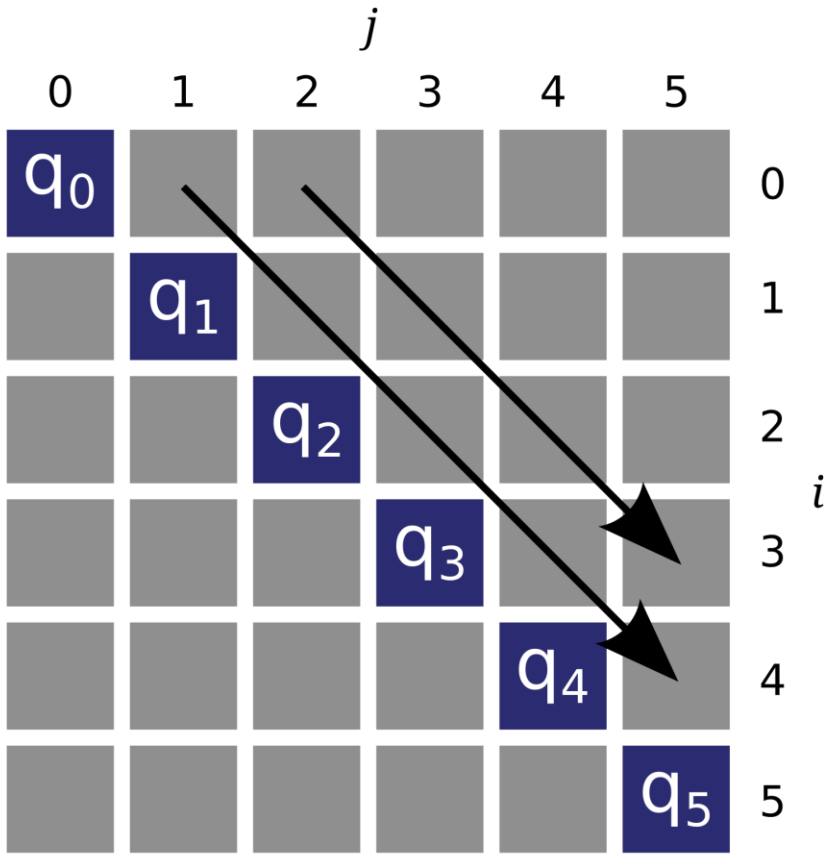
Algorithm: Dynamic Programming

- per entry
  - h additions
  - h comparisions
- $\mathrm{O}(n^2)$ entries

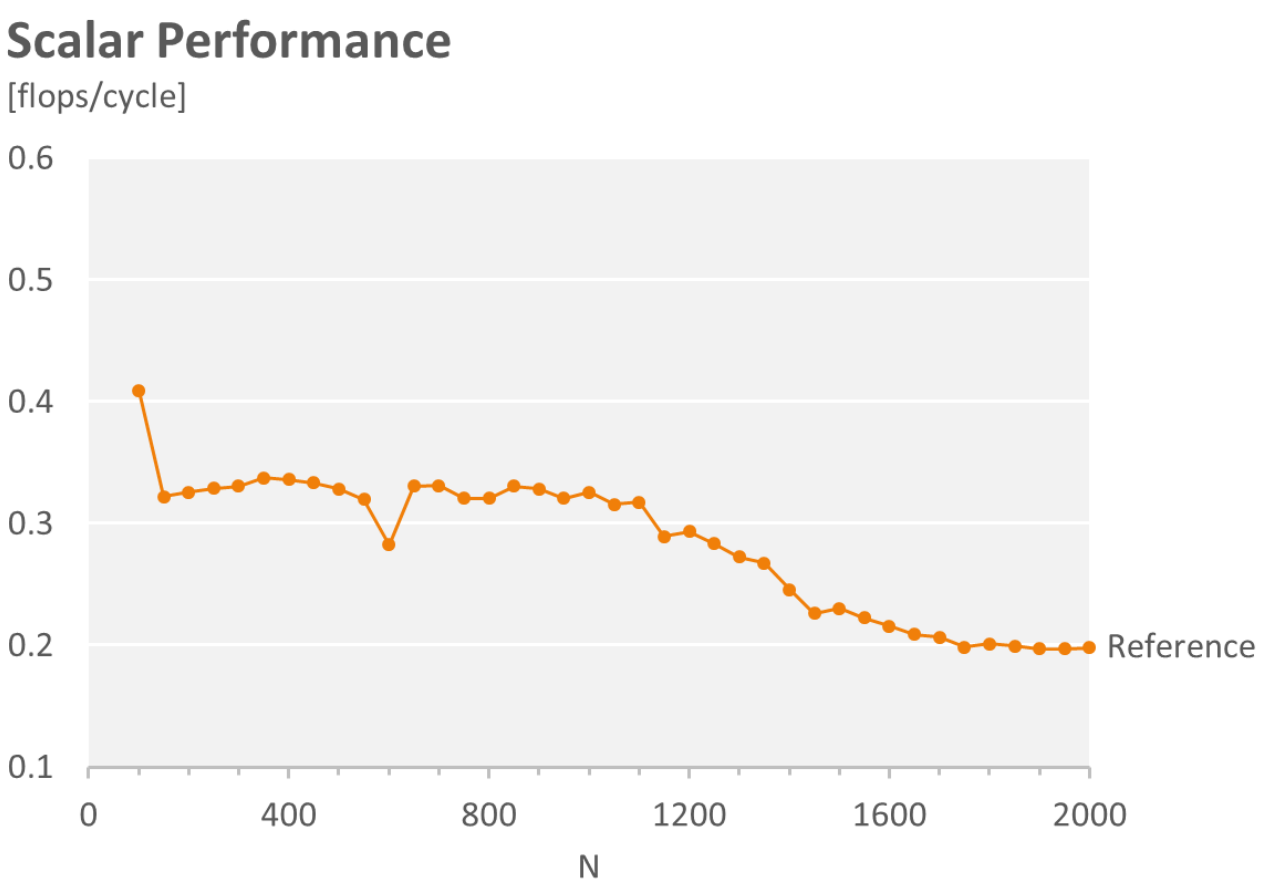


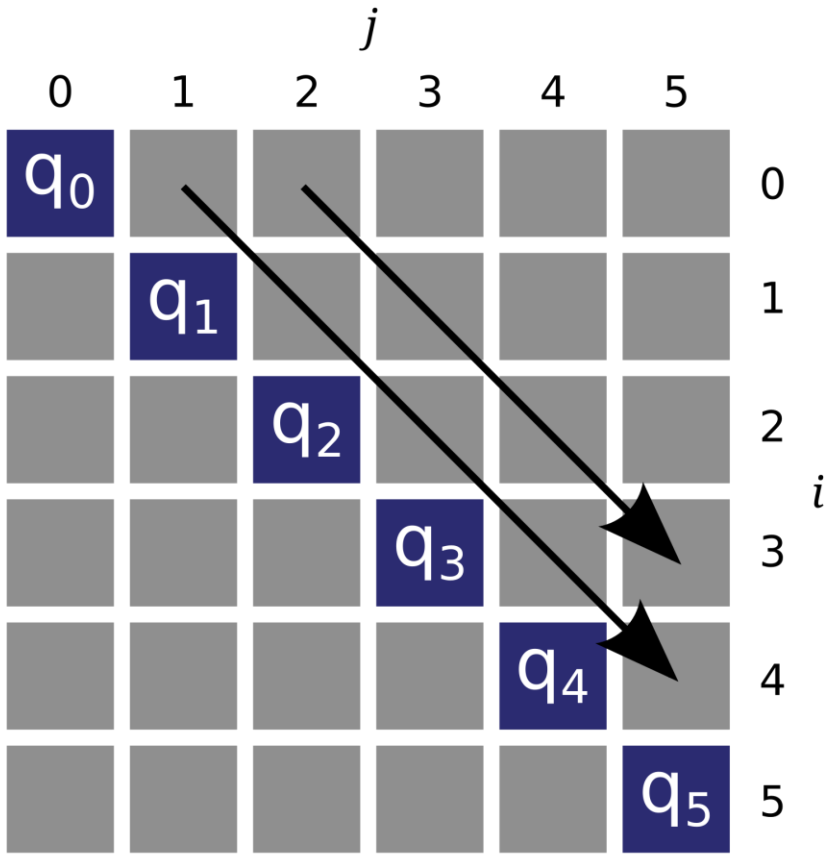
- $\mathrm{C}(\mathrm{add}, \mathrm{cmp}) = \mathrm{C}(\frac{1}{3}n^3 + \mathrm{O}(\mathrm{n}^2), \frac{1}{6}n^3 + \mathrm{O}(\mathrm{n}^2))$

Reference Implementation

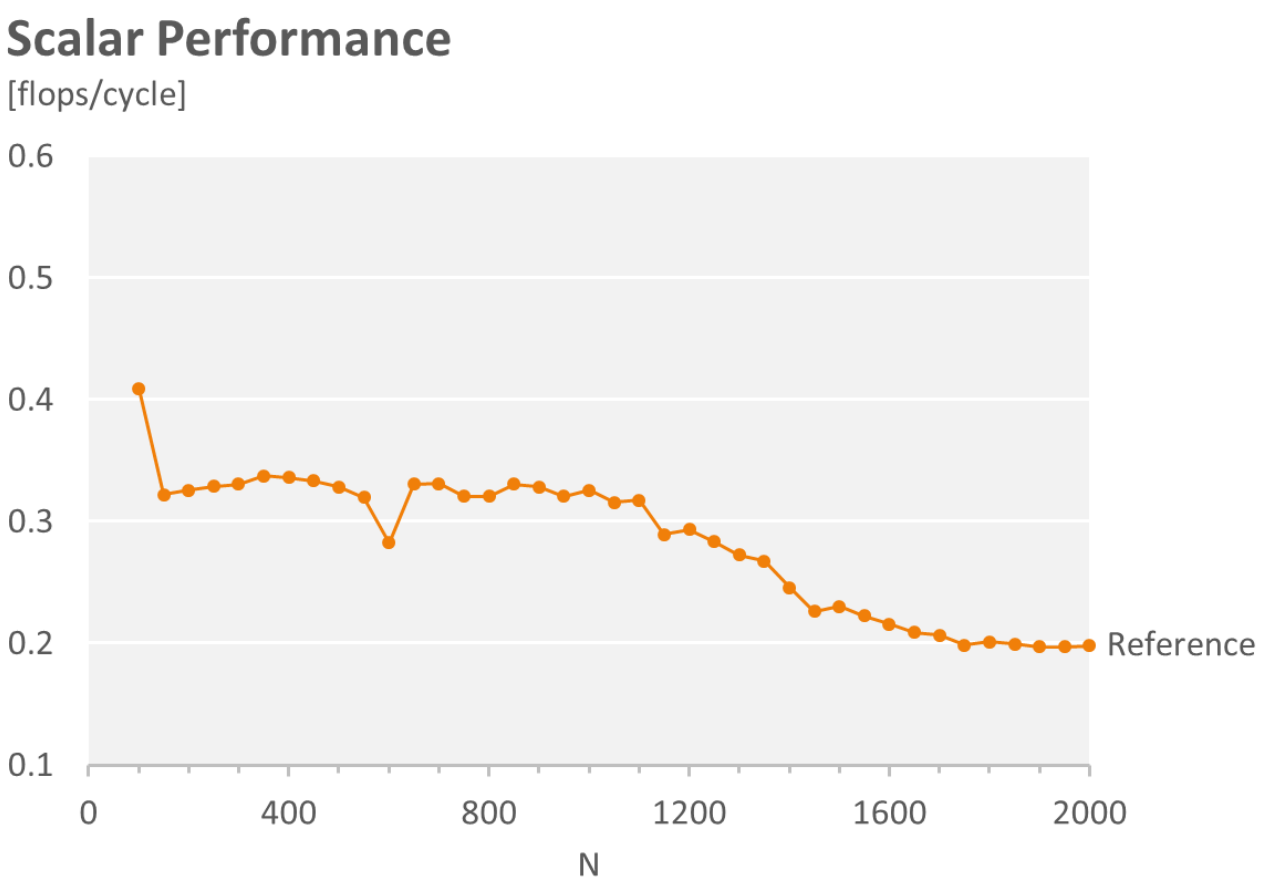


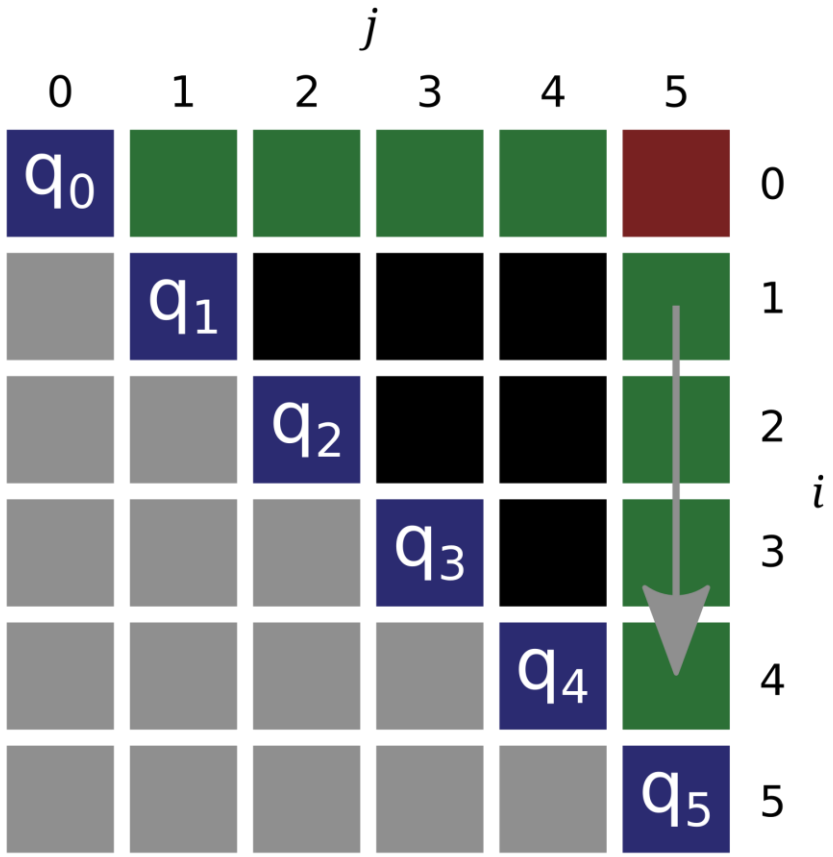
Reference Implementation



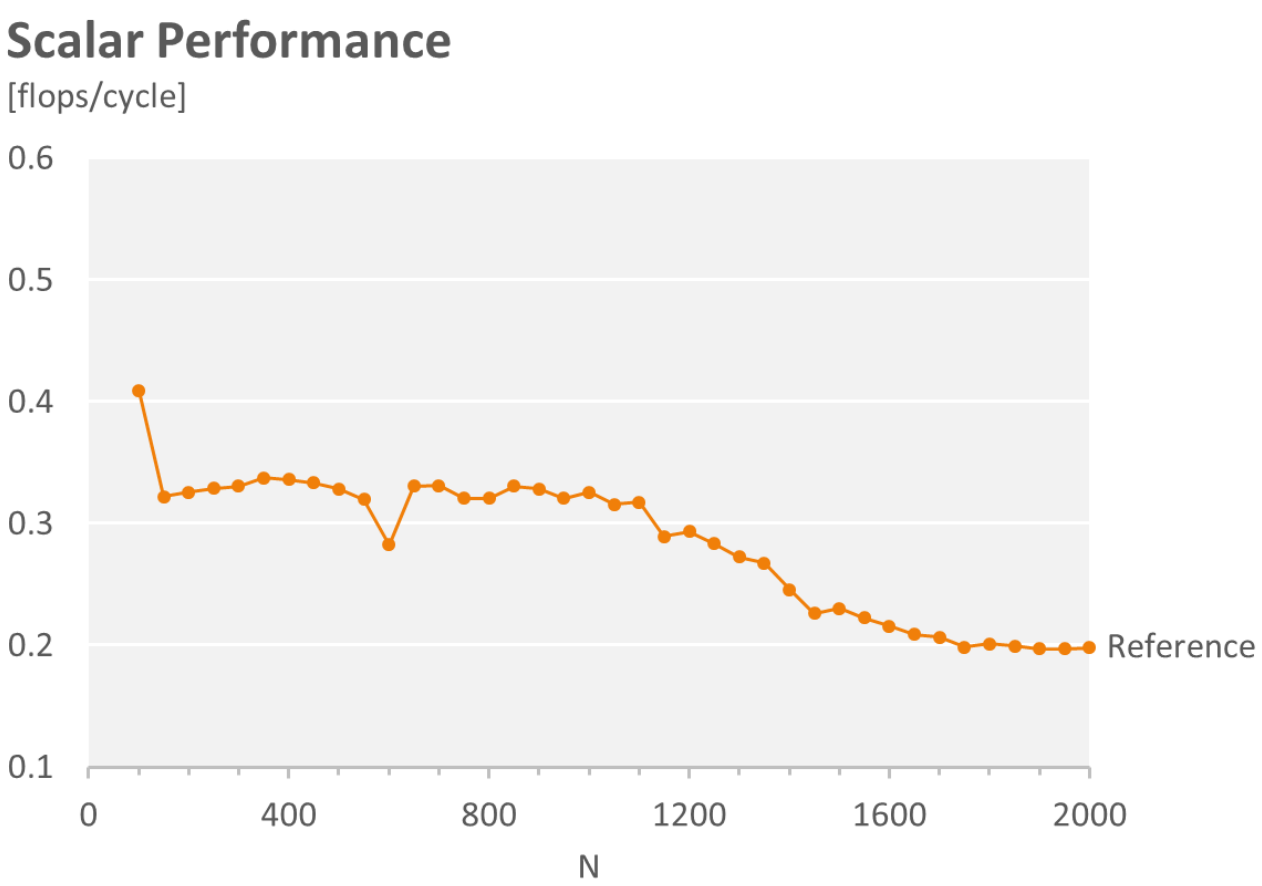


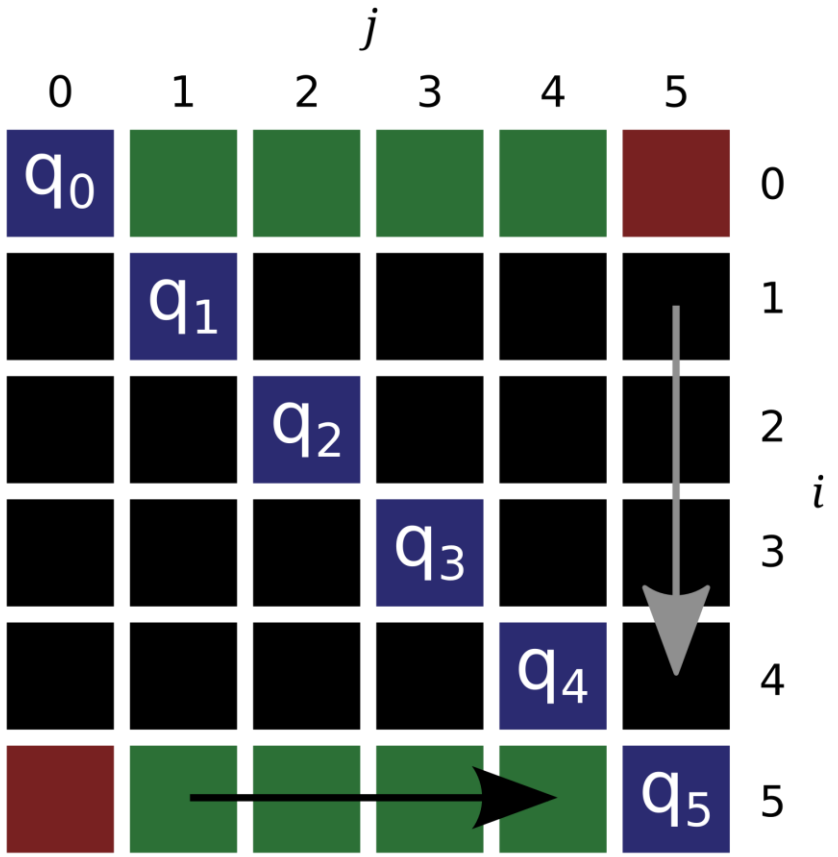
Reference Issues





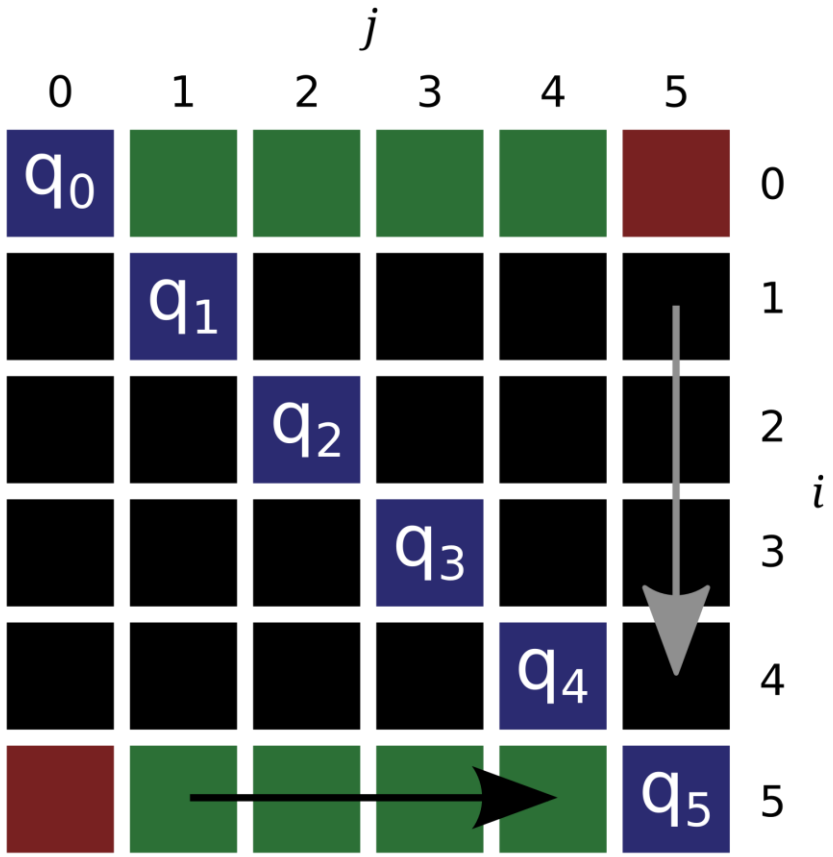
Transposed


Scalar Performance
[flops/cycle]
0.6
0.5
0.4
0.3
0.2
0.1
0
400
800
1200
1600
2000
N
Reference


j
0
1
2
3
4
5
i
q_0
q_1
q_2
q_3
q_4
q_5

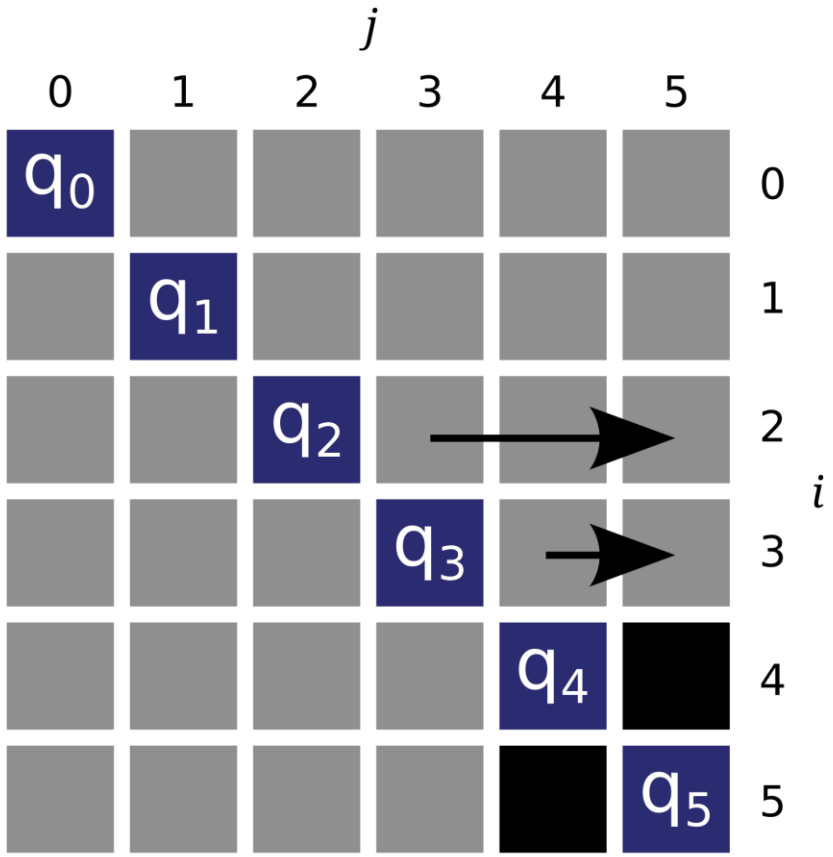
Transposed


Scalar Performance
[flops/cycle]
0.6
0.5
0.4
0.3
0.2
0.1
0
400
800
1200
1600
2000
N
Transposed
Reference


j
0
1
2
3
4
5
i
0
1
2
3
4
5
q_0
q_1
q_2
q_3
q_4
q_5

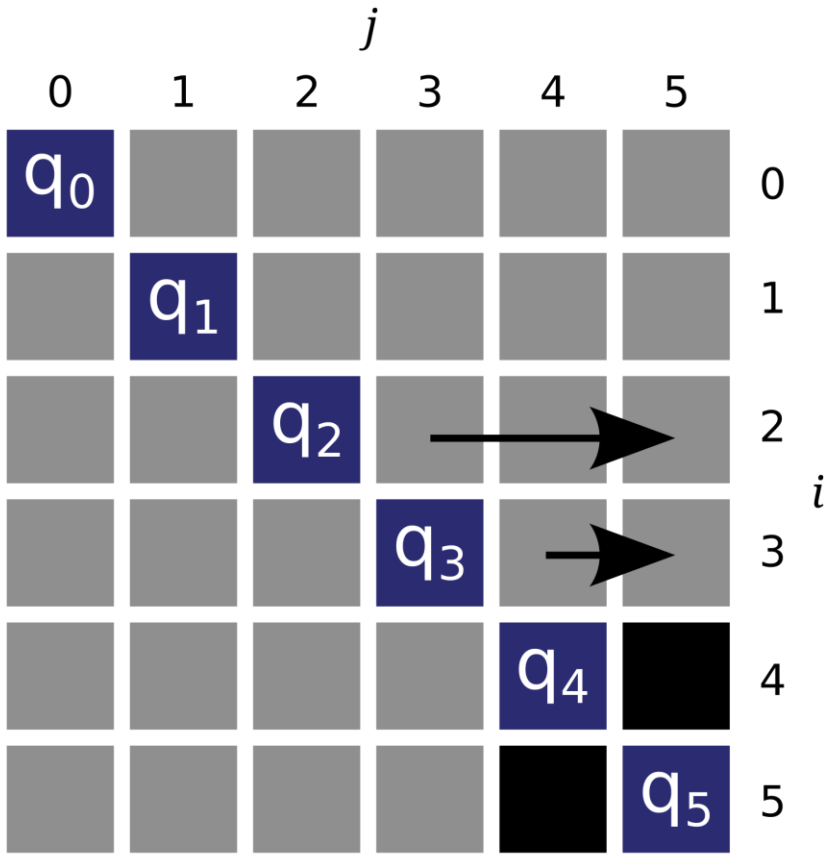
Bottom Up


Scalar Performance
[flops/cycle]
0.6
0.5
0.4
0.3
0.2
0.1
0
400
800
1200
1600
2000
N
Transposed
Reference

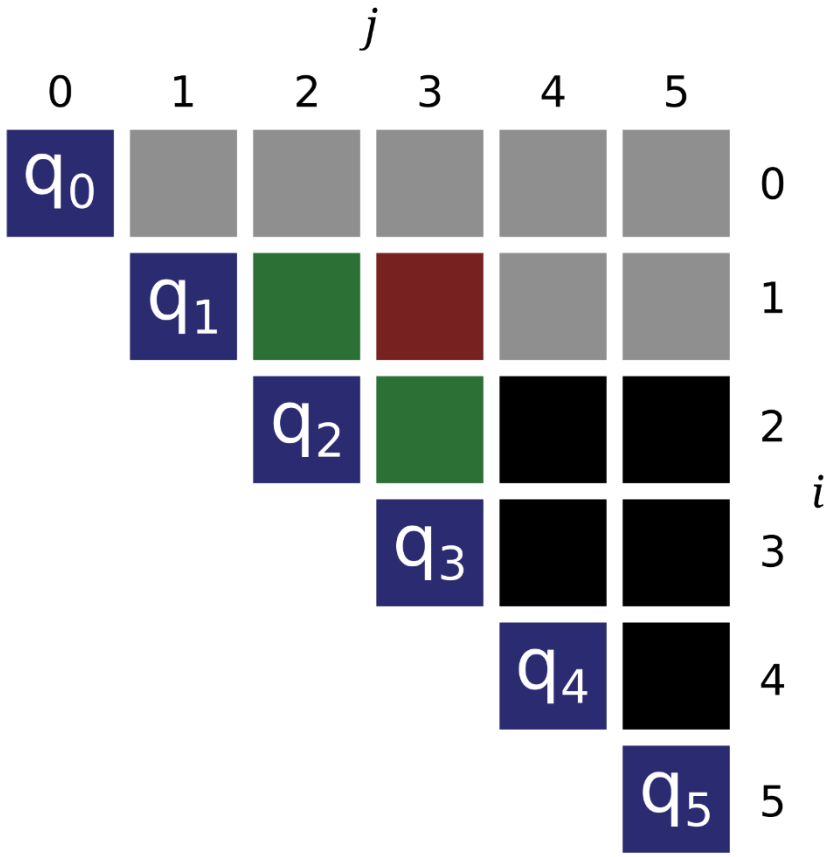

j
0
1
2
3
4
5
i
0
1
2
3
4
5
q_0
q_1
q_2
q_3
q_4
q_5

Bottom Up

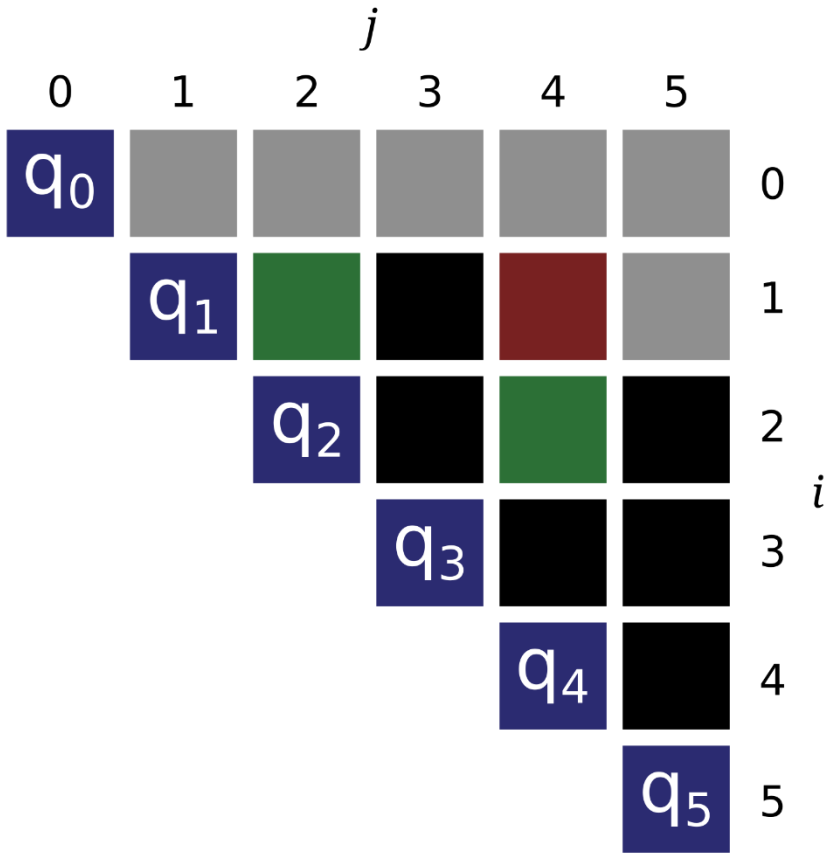

Scalar Performance
[flops/cycle]
0.6
0.5
0.4
0.3
0.2
0.1
0
400
800
1200
1600
2000
N
Bottom Up
Transposed
Reference


j
0
1
2
3
4
5
i
q_0
q_1
q_2
q_3
q_4
q_5

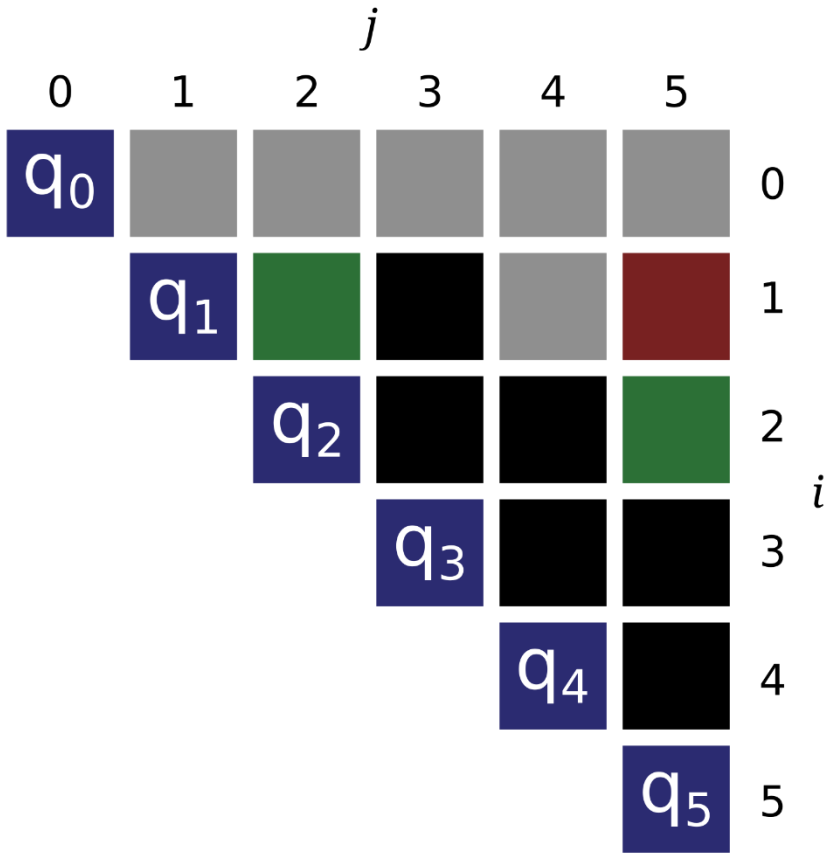
Swap Loops: Triangle



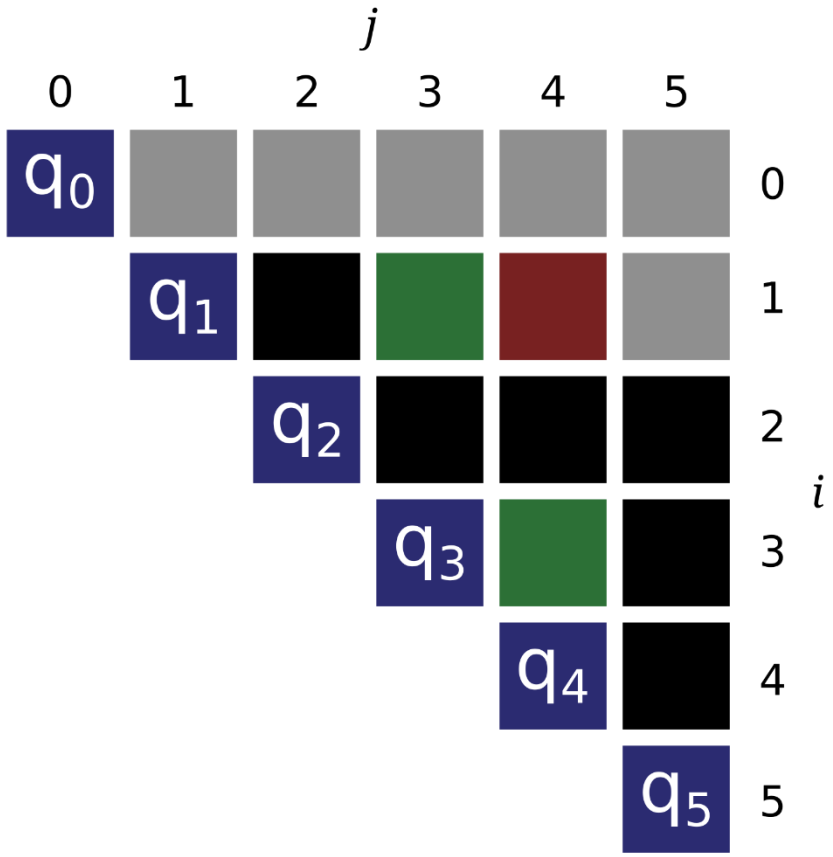
Swap Loops: Triangle



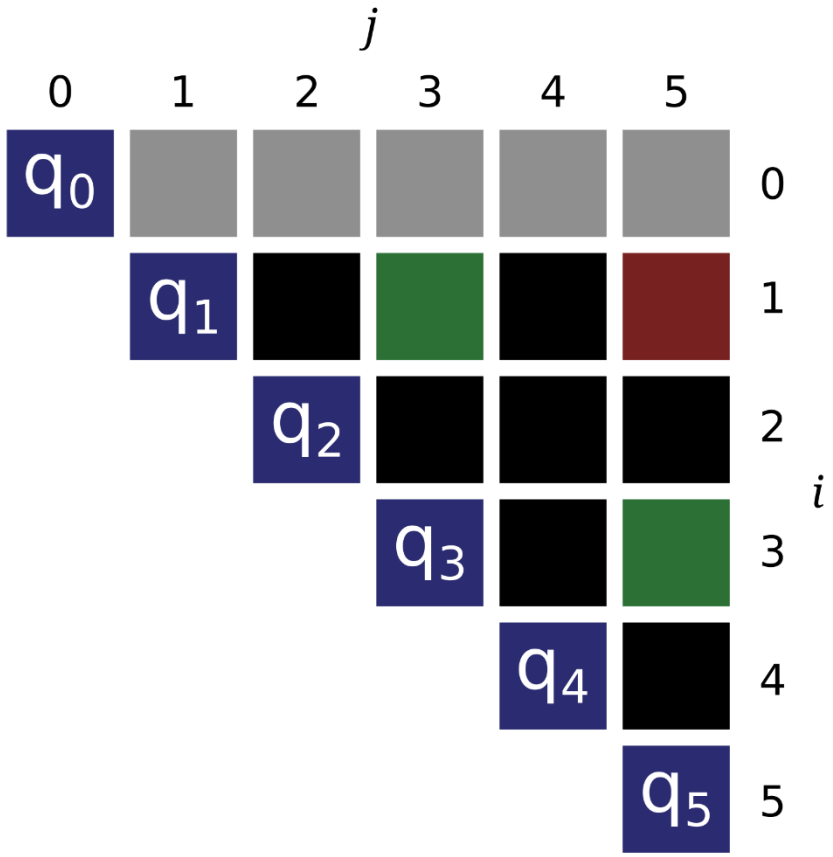
Swap Loops: Triangle



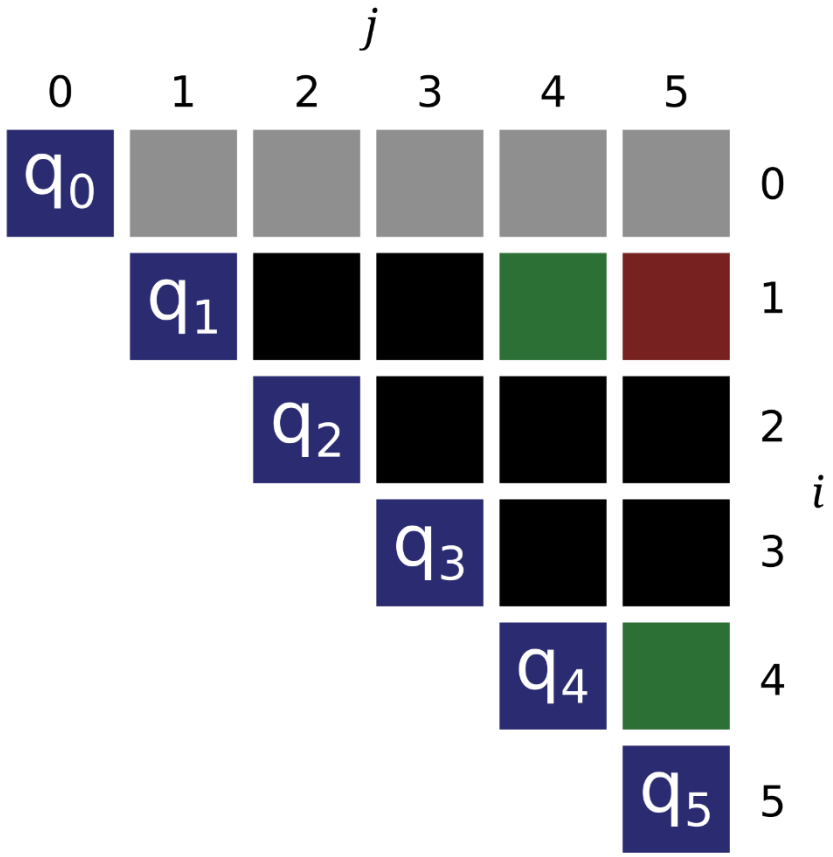
Swap Loops: Triangle



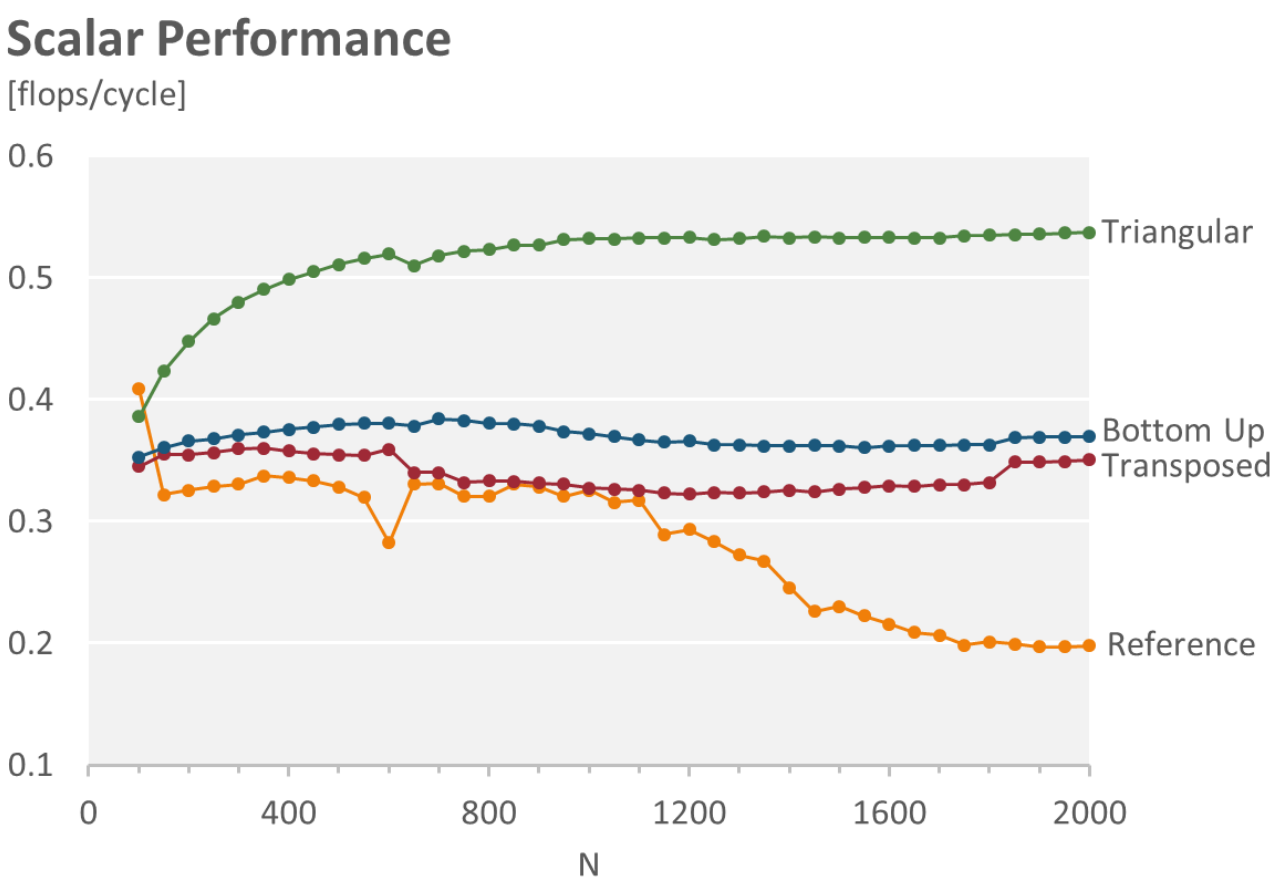
Swap Loops: Triangle

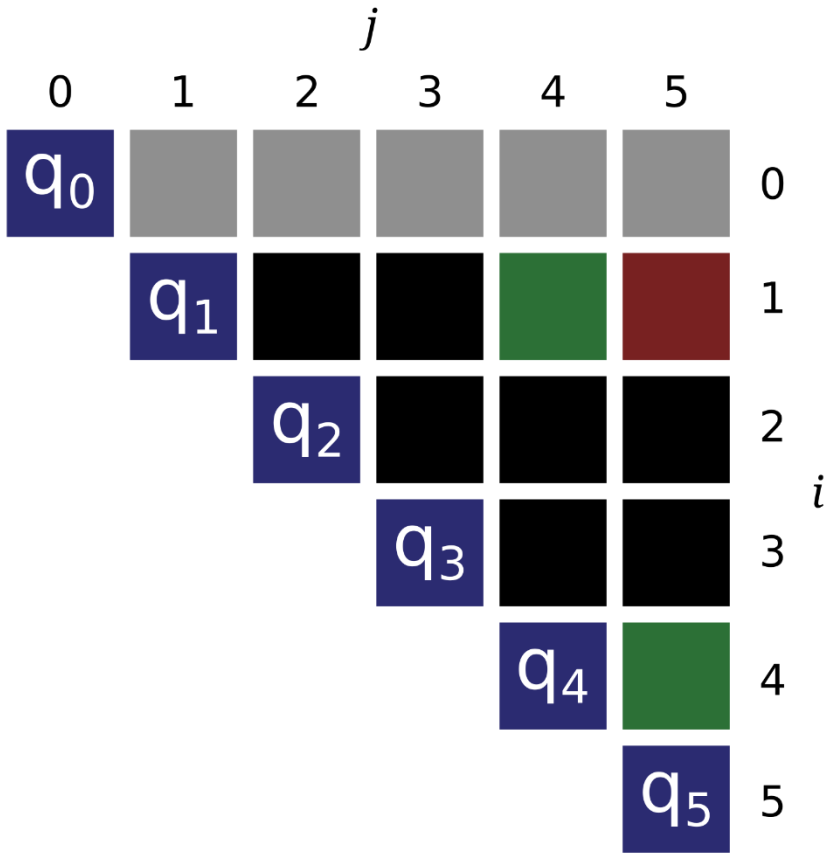


Swap Loops: Triangle

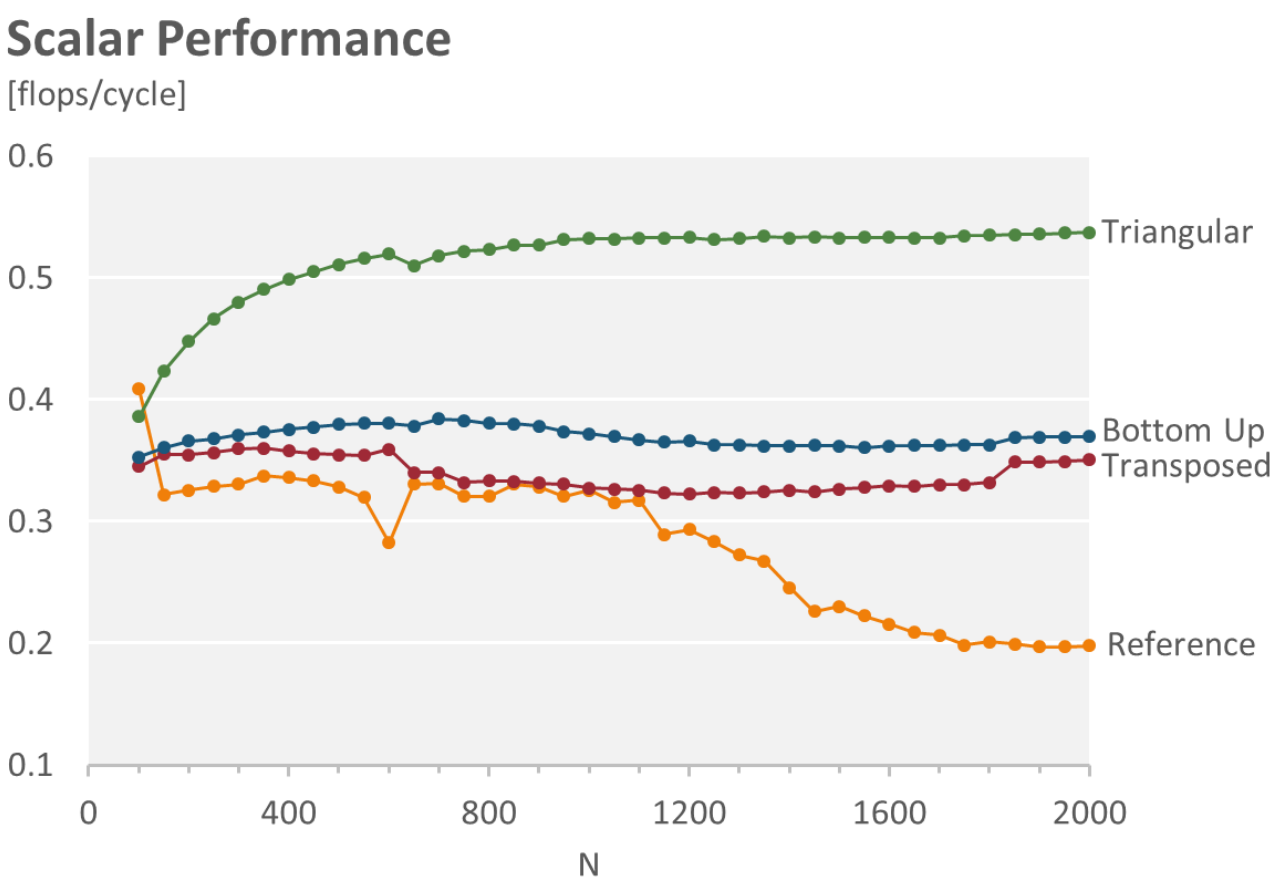


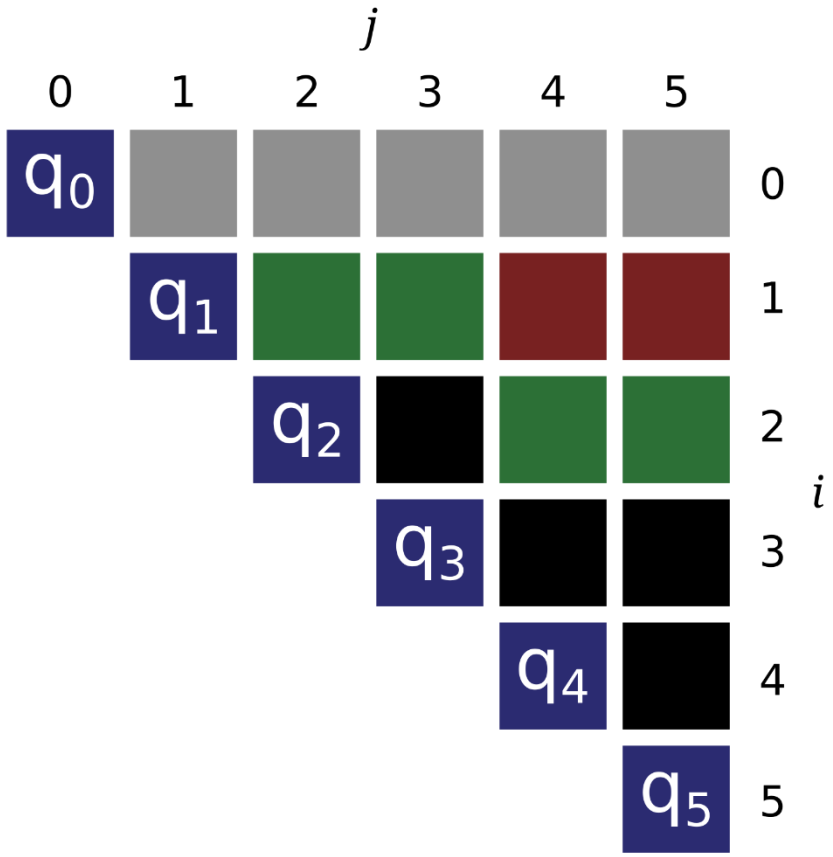
Swap Loops: Triangle



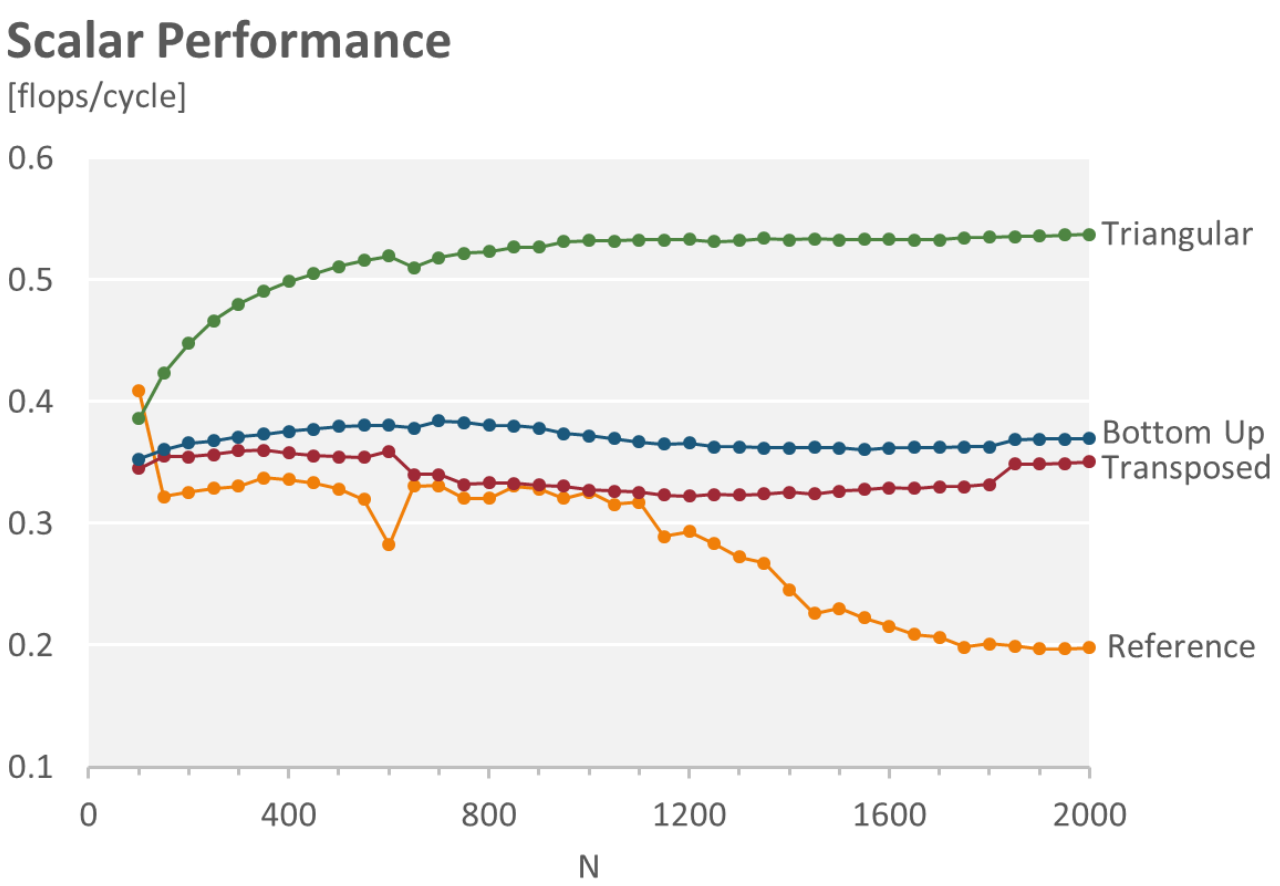


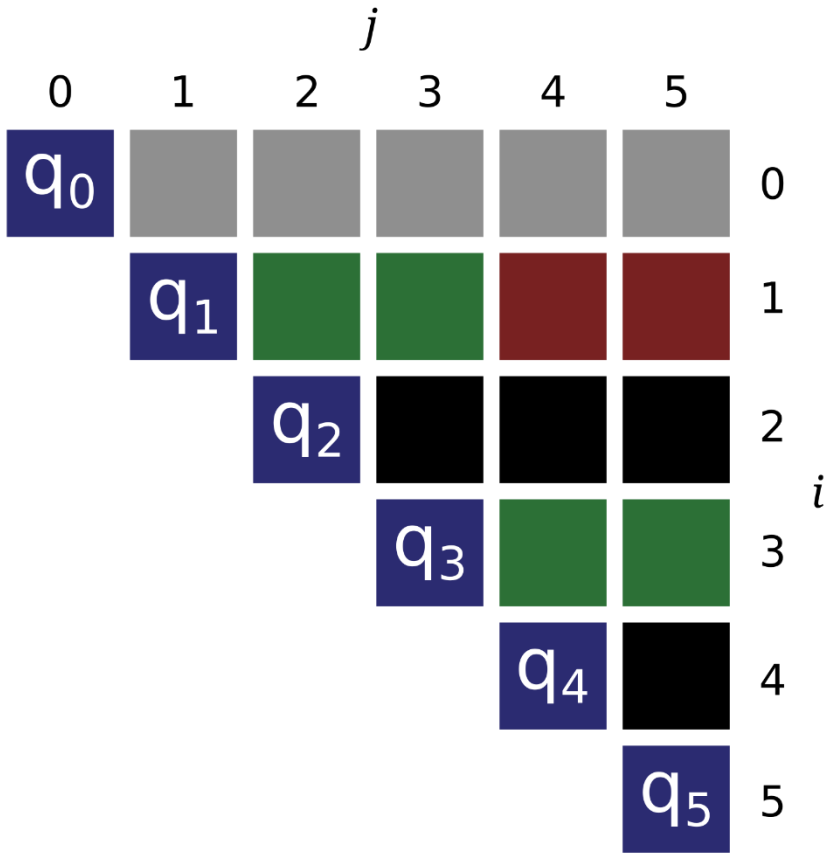
Blocking


Scalar Performance
[flops/cycle]
0.6
0.5
0.4
0.3
0.2
0.1
0
400
800
1200
1600
2000
N
Triangular
Bottom Up
Transposed
Reference

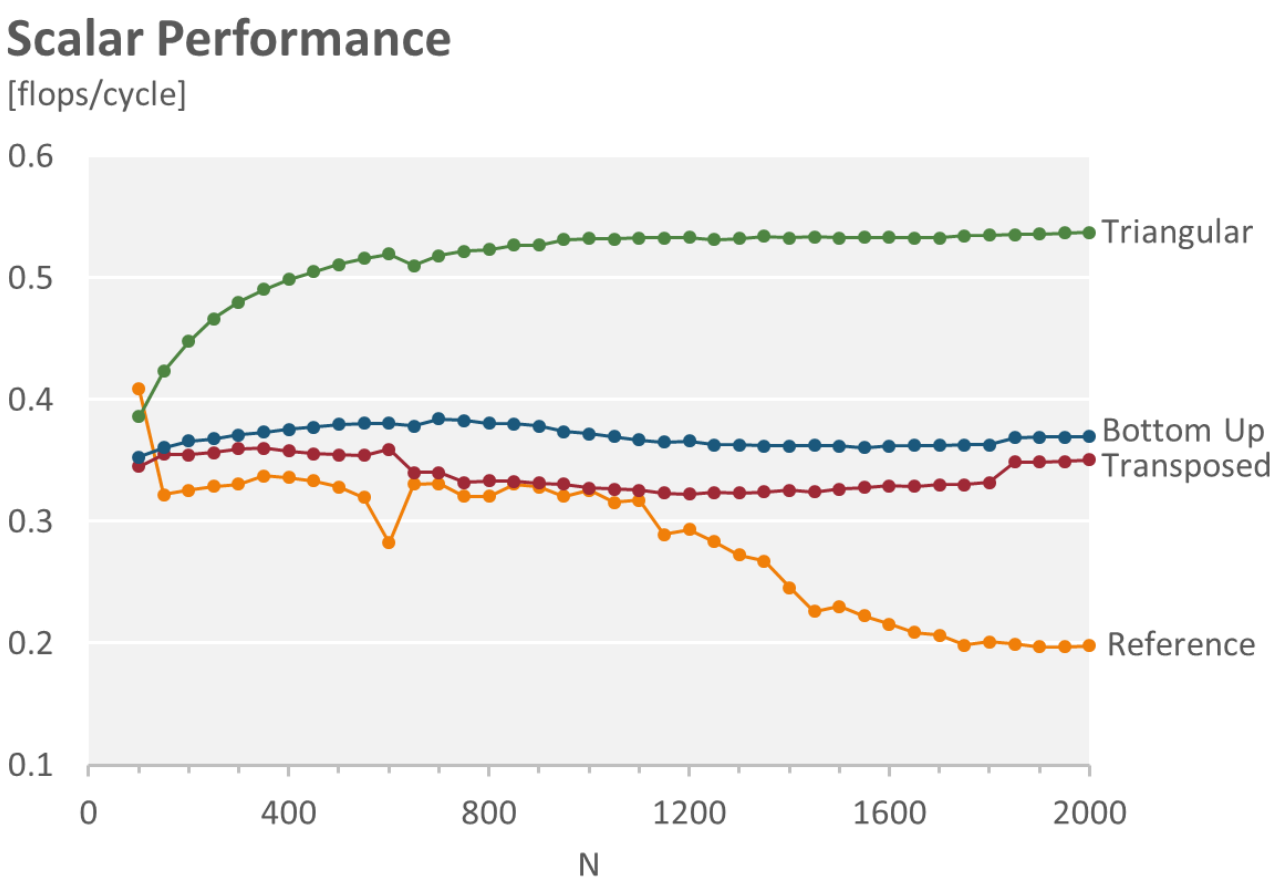

j
0
1
2
3
4
5
i
q_0
q_1
q_2
q_3
q_4
q_5

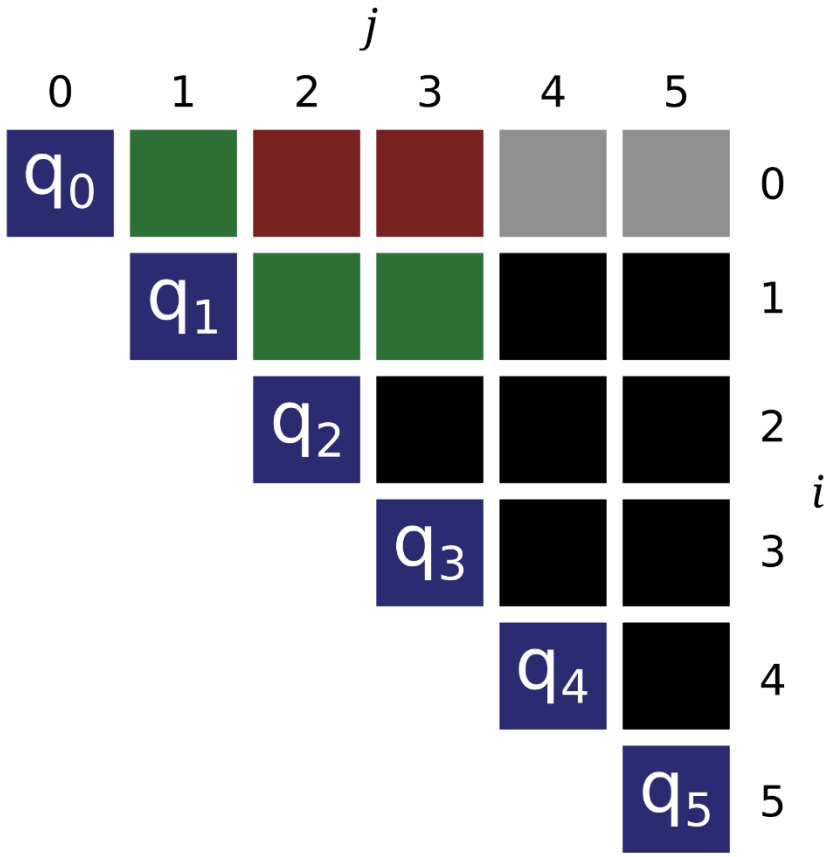
Blocking


Scalar Performance
[flops/cycle]
0.6
0.5
0.4
0.3
0.2
0.1
0
400
800
1200
1600
2000
N
Triangular
Bottom Up
Transposed
Reference

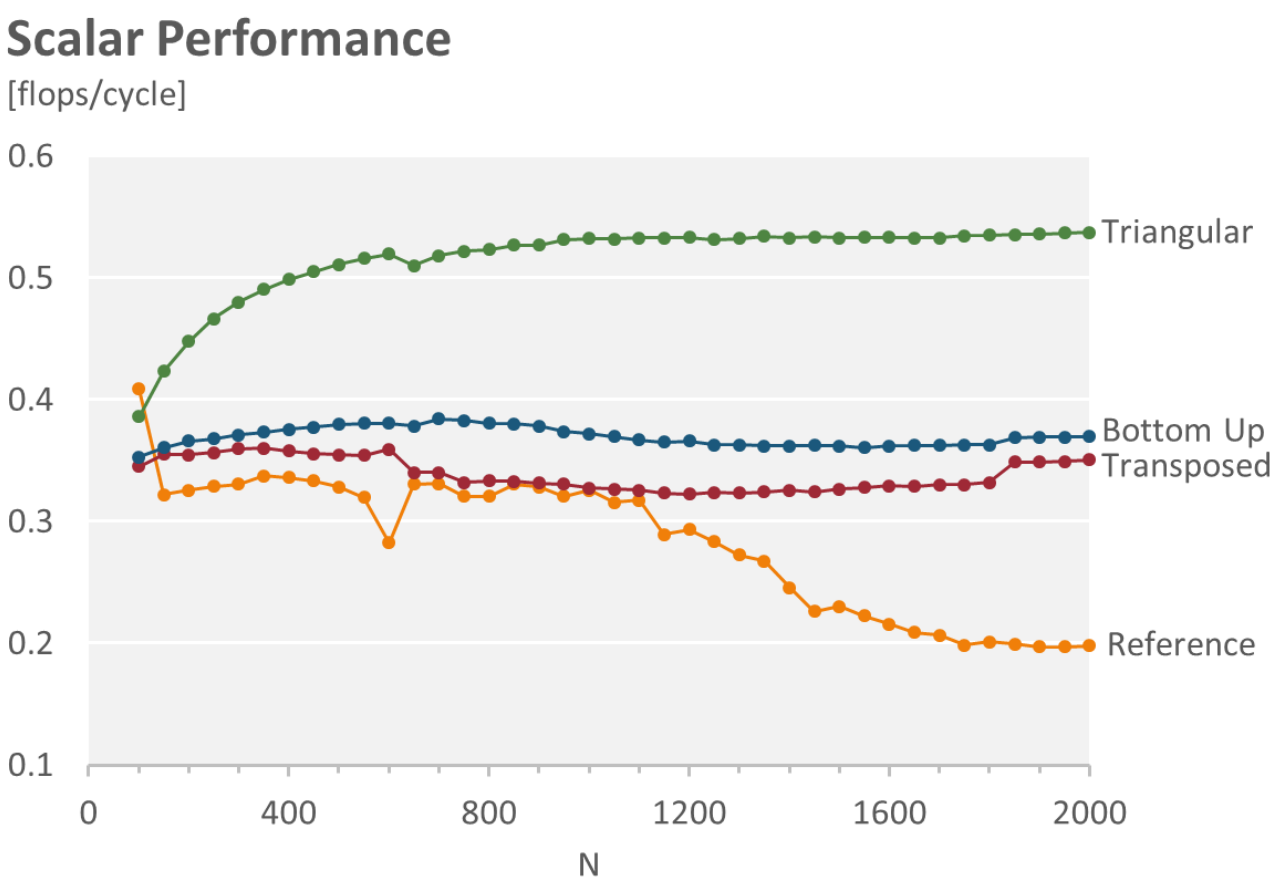

j
0
1
2
3
4
5
i
q_0
q_1
q_2
q_3
q_4
q_5

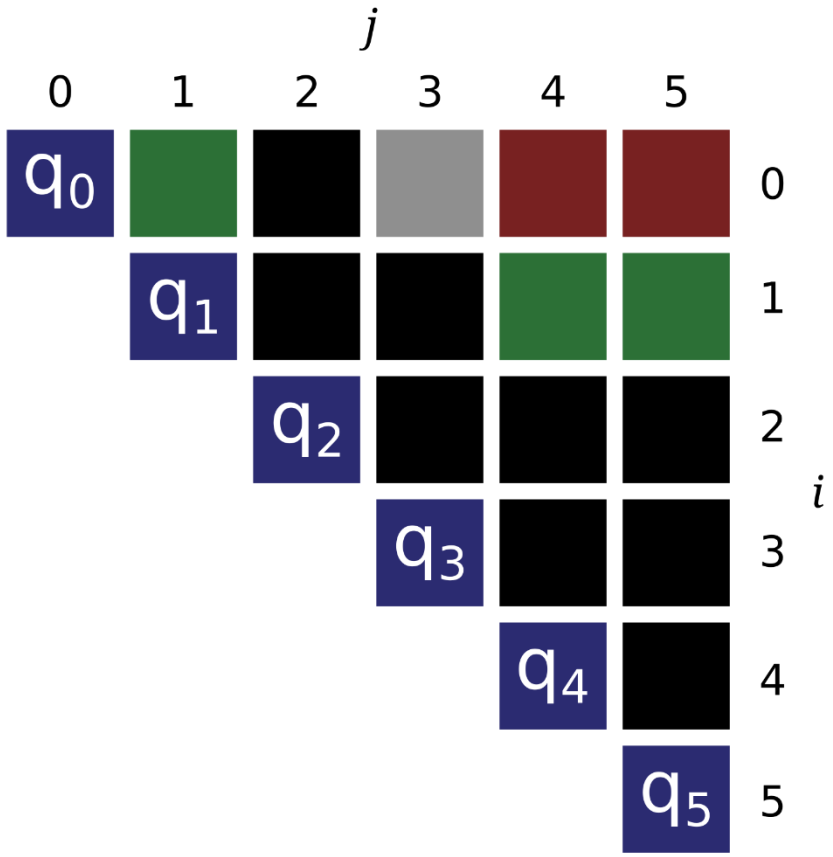
Vectorization



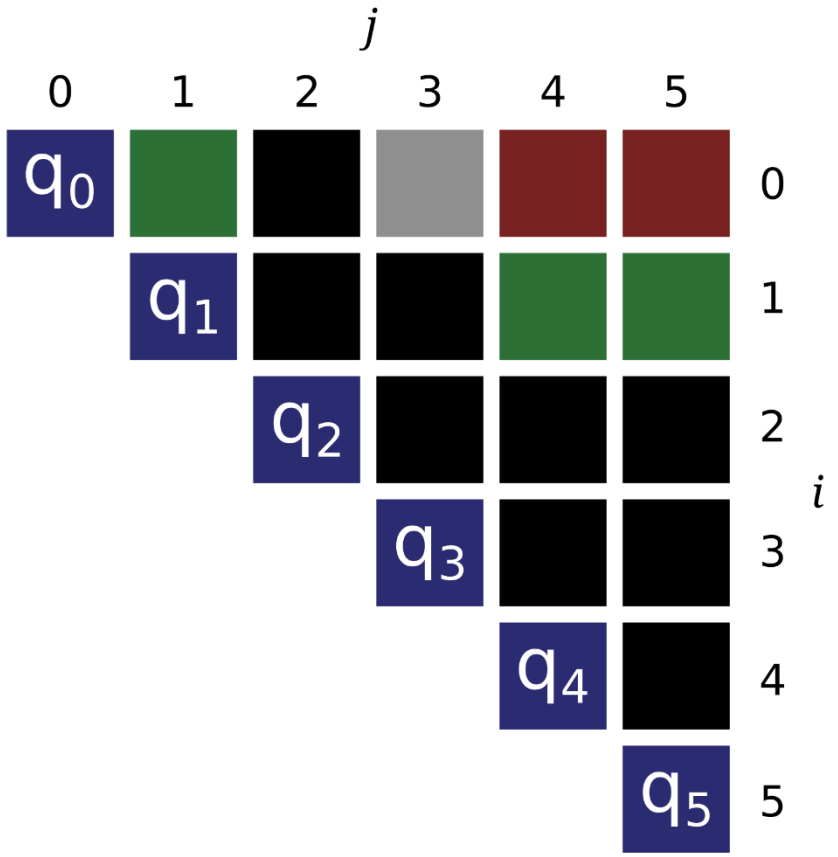


Vectorization





Vectorization



Vectorization Issues: Scheduling

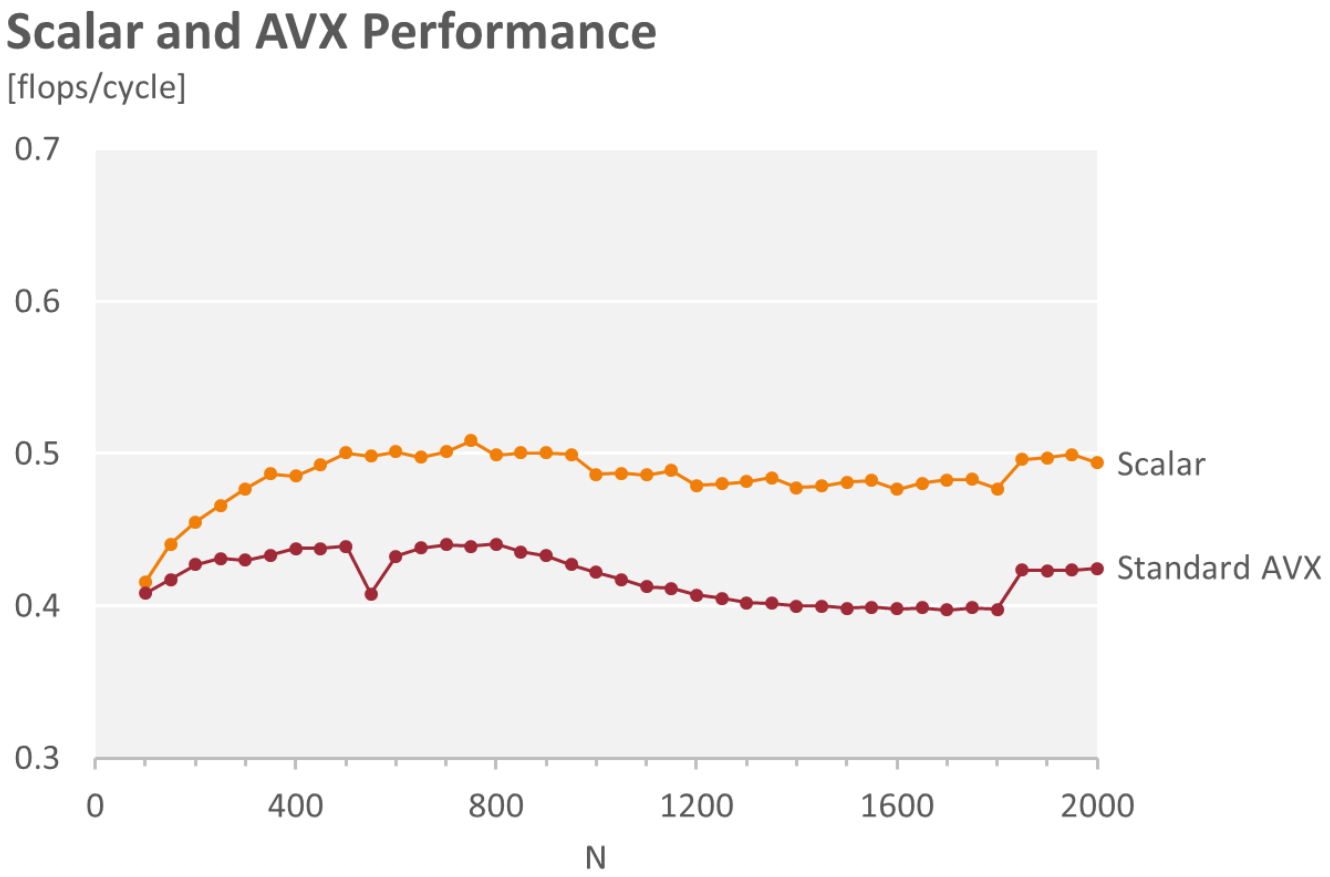
Standard

```
a = load()
b = load()
c = load()
d = load()

e = add(a,b)
f = add(c,d)
g = add(e,f)

store(g)
```

Vectorization Issues: Scheduling

Standard

```
a = load()
b = load()
c = load()
d = load()

e = add(a,b)
f = add(c,d)
g = add(e,f)

store(g)
```



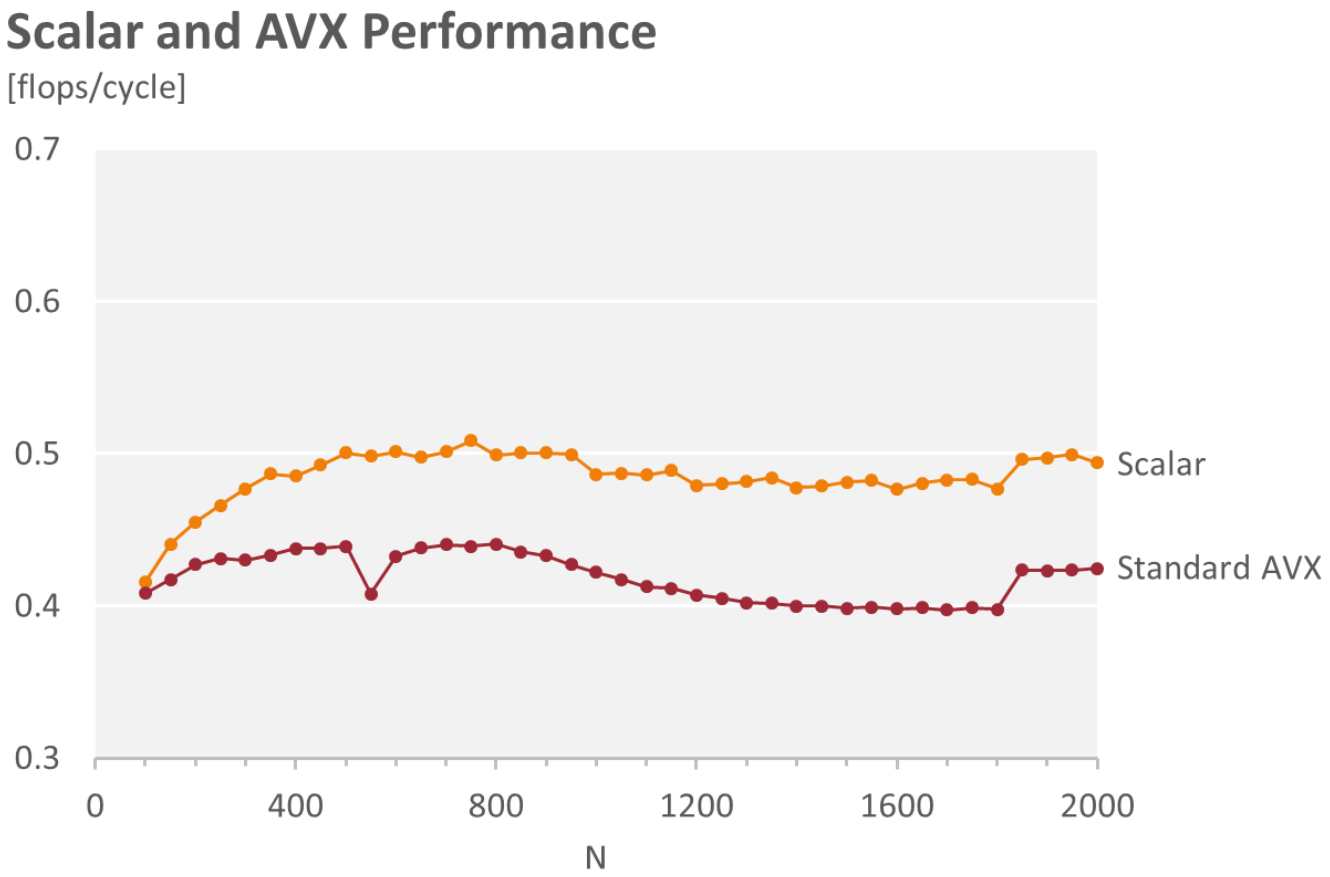
Vectorization Issues: Scheduling

Standard

```
a = load()
b = load()
c = load()
d = load()

e = add(a,b)
f = add(c,d)
g = add(e,f)

store(g)
```

Rescheduled

```
a = load()
b = load()
e = add(a,b)

c = load()
d = load()
f = add(c,d)

g = add(e,f)
store(g)
```



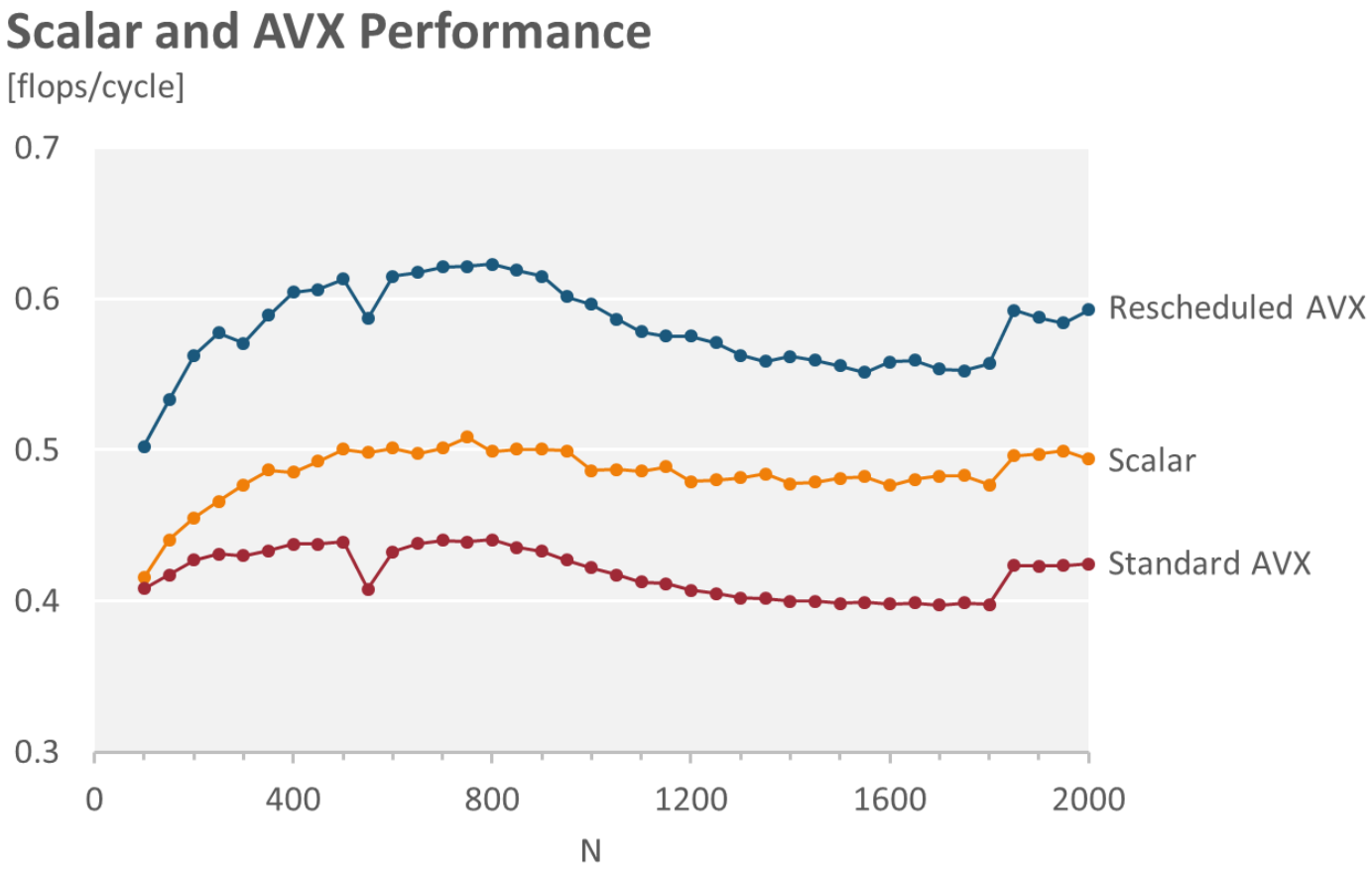
Vectorization Issues: Scheduling

Standard

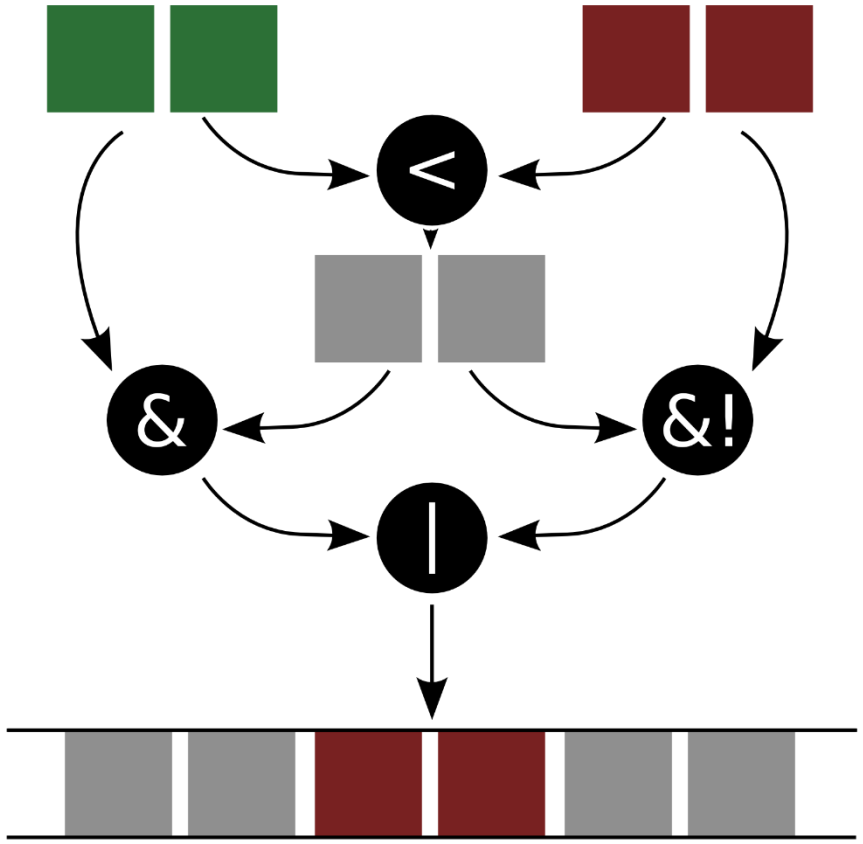
```
a = load()
b = load()
c = load()
d = load()

e = add(a,b)
f = add(c,d)
g = add(e,f)

store(g)
```

Rescheduled

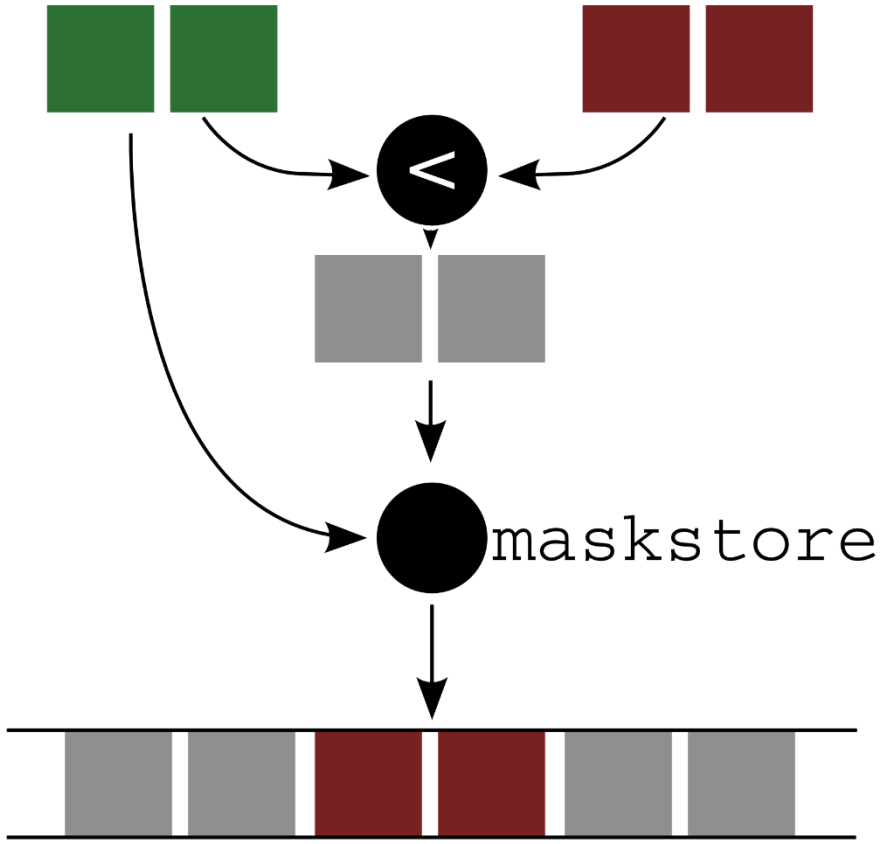
```
a = load()
b = load()
e = add(a,b)

c = load()
d = load()
f = add(c,d)

g = add(e,f)
store(g)
```



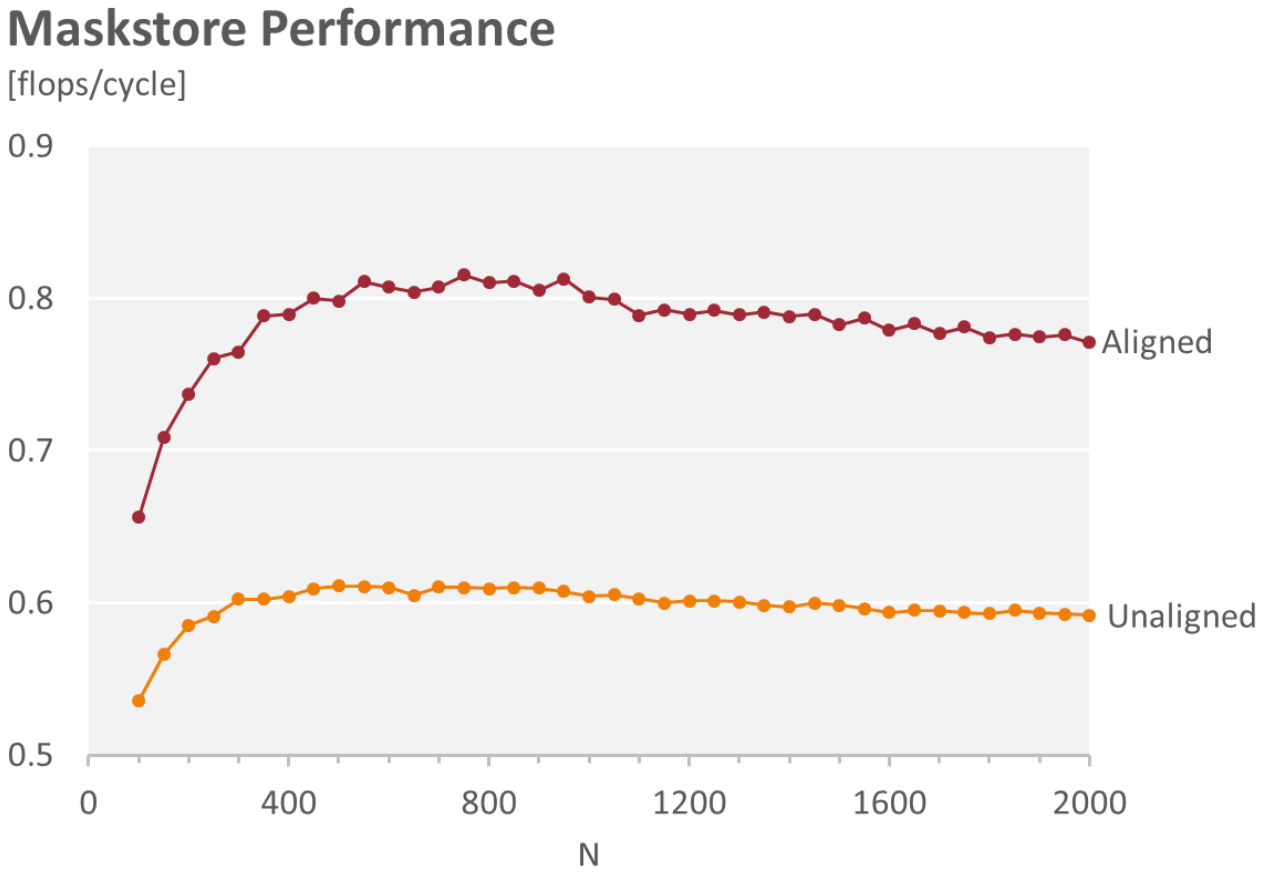
Vectorization Issues: Maskstore

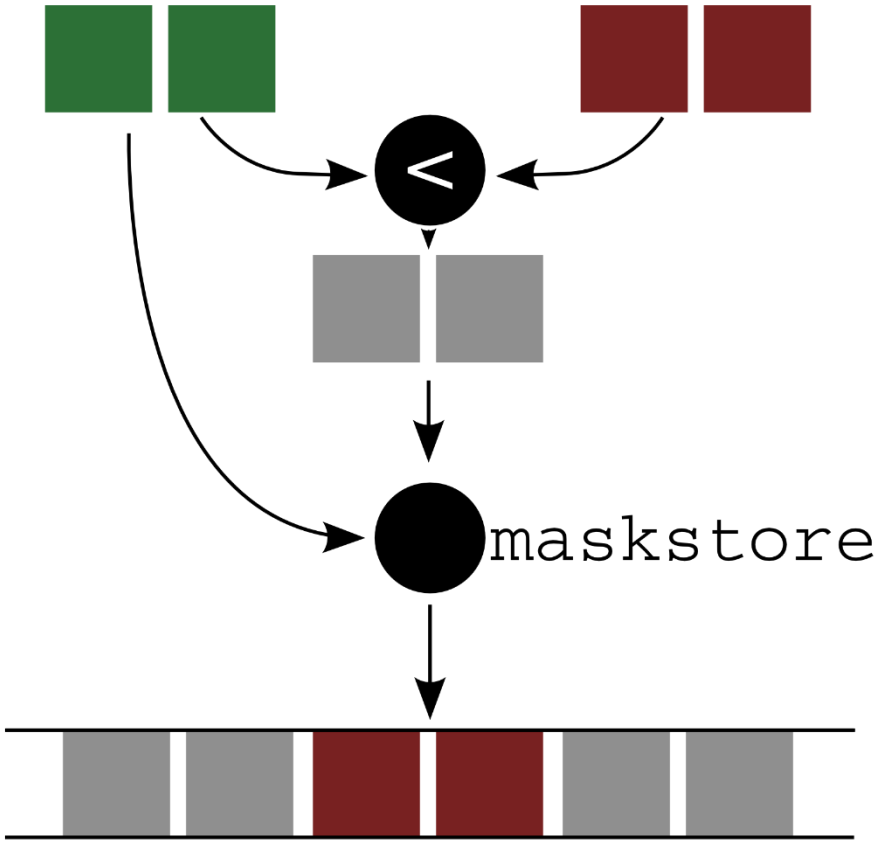


Vectorization Issues: Maskstore

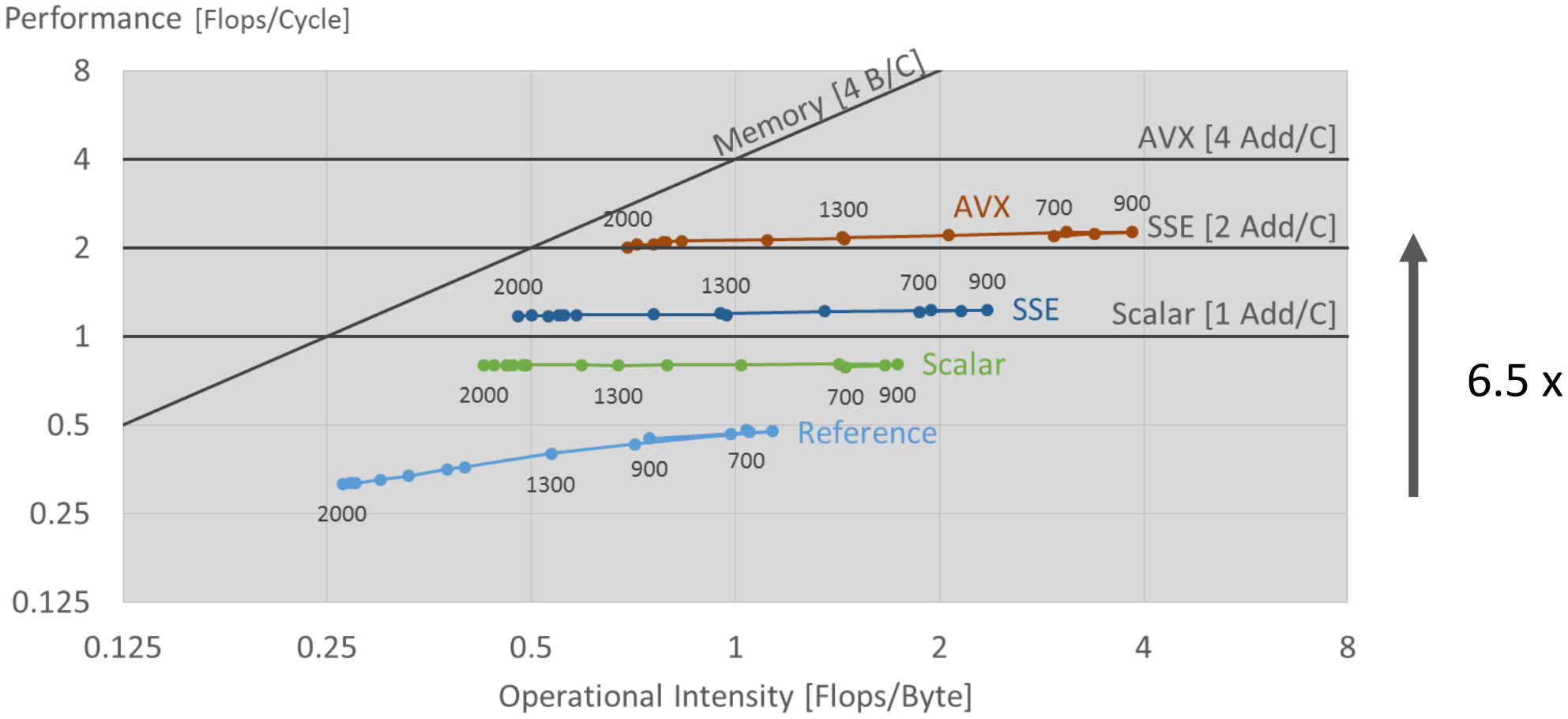


Vectorization Issues: Maskstore





Roofline Plot



6.5 x

Questions…?