Computational Science on Many-Core Architectures

360.252

Karl Rupp

Institute for Microelectronics
Vienna University of Technology
http://www.iue.tuwien.ac.at

Zoom Channel 95028746244
Wednesday, November 18, 2020

Agenda for Today

Exercise 4 Recap

Finite Differences

Recap: Prefix Sums

Sequential to Parallel: ILU Example

Exercise 5

Exercise 4 Recap

Feedback Time

- How was your experience?

Exercise 4 Recap

Feedback Time

- How was your experience?
- You should have received links to points for course last week

Exercise 4 Recap

Feedback Time

- How was your experience?
- You should have received links to points for course last week

Vector Iterations

```
__kernel__ void option1(double *x, int N, ...) {
for (int row = blockDim.x * blockIdx.x + threadIdx.x;
      row < N;
      row += gridDim.x * blockDim.x) { ... }
}
option1<<<256, 256>>>(x, N, ...);  // arbitrary grid/block dim
```

```
__kernel__ void option2(double *x, int N, ...) {
  if (blockDim.x*blockIdx.x + threadIdx.x < N) { ... }
}
option2<<<N/blocksize + 1, blocksize>>>(x, N, ...); // beware!
```

Finite Differences

Discrete Poisson Equation

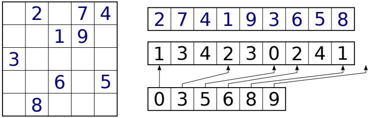
- Rectangular grid
- Assume homogeneous grid spacing h
- Discretize each coordinate direction separately

$$
\begin{aligned}
\Delta u(x,y) &= u_{xx}(x,y) + u_{yy}(x,y) \\
&\approx \frac{u(x-h,y) - 2u(x,y) + u(x+h,y)}{h^2} + \frac{u(x,y-h) - 2u(x,y) + u(x,y+h)}{h^2} \\
&= \frac{u(x-h,y) + u(x+h,y) - 4u(x,y) + u(x,y-h) + u(x,y+h)}{h^2} \\
&=: \Delta_h u(x,y)
\end{aligned}
$$

Finite Differences

System Matrix Structure

- Interior nodes have 5 nonzero entries
- Boundary nodes have 1 to 4 nonzero entries
- Problem: CSR format requires offset index for each row

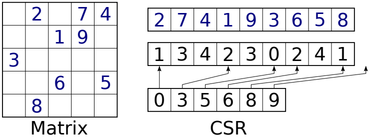


Matrix CSR

Finite Differences

System Matrix Structure

- Interior nodes have 5 nonzero entries
- Boundary nodes have 1 to 4 nonzero entries
- Problem: CSR format requires offset index for each row



Generic GPU matrix assembly skeleton

1. Count the nonzero entries for each row
2. Deduce offsets for CSR
3. Populate column and values arrays

Finite Differences

Step 1: Count Nonzeros for an $N \times M$ grid

```
__kernel__ void count_nnz(int *row_offsets, int N, ...) {
for (int row = blockDim.x * blockIdx.x + threadIdx.x;
      row < N*M;
      row += gridDim.x * blockDim.x)
    {
        int nnz_for_this_node = 1;
        int i = row / N;
        int j = row % N;

        if (i > 0) nnz_for_this_node += 1;
        if (j > 0) nnz_for_this_node += 1;
        if (i < N-1) nnz_for_this_node += 1;
        if (j < M-1) nnz_for_this_node += 1;

        row_offsets[row] = nnz_for_this_node;
      }
}
```

Parallel Primitives

Prefix Sum

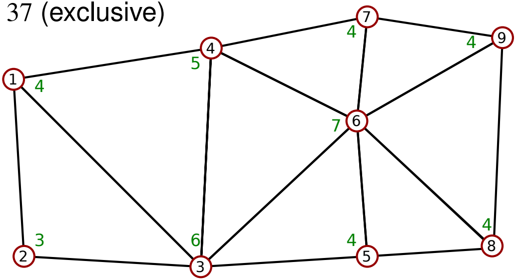
- Inclusive: Determine $y_i = \sum_{k=1}^{i} x_k$
- Exclusive: Determine $y_i = \sum_{k=1}^{i-1} x_k$, $y_1 = 0$

Example

- x: 4, 3, 6, 5, 4, 7, 4, 4, 4
- y: 4, 7, 13, 18, 22, 29, 33, 37, 41 (inclusive)
- y: 0, 4, 7, 13, 18, 22, 29, 33, 37 (exclusive)

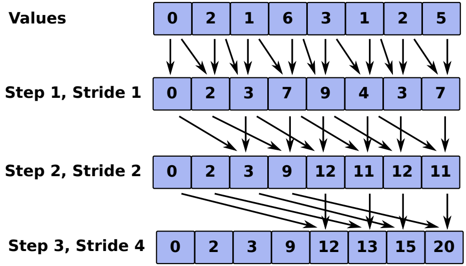
Applications

- Sparse matrix setup
- Graph algorithms



Parallel Primitives

Prefix Sum Implementation




```
for(int stride = 1; stride < blockDim.x; stride *= 2)
{
  __syncthreads();
  shared_m[threadIdx.x] = my_value;
  __syncthreads();
  if (threadIdx.x >= stride)
    my_value += shared_m[threadIdx.x - stride];
}
__syncthreads();
shared_m[threadIdx.x] = my_value;
```

Finite Differences

Step 3: Assembly for an $N \times M$ grid

```
__kernel__ void assembleA(int *row_offsets, int N, ...) {
for (int row = blockDim.x * blockIdx.x + threadIdx.x;
      row < N*M;
      row += gridDim.x * blockDim.x) {
        int i = row / N;
        int j = row % N;
        int this_row_offset = row_offsets[row];

        int index = i * N + j;
        col_indices[this_row_offset] = index;
        values[this_row_offset] = 4;
        this_row_offset += 1;

        if (i > 0) { /* similarly with correct index and value -1
            */ }
        if (j > 0) { /* similarly */ }
        if (i < N-1)  { /* similarly */ }
        if (j < M-1) { /* similarly */ }
      }
}
```

Other Discretizations

Finite Volumes

- Iterations over vertices: Similar to finite differences
- Beware of advanced schemes (depends)

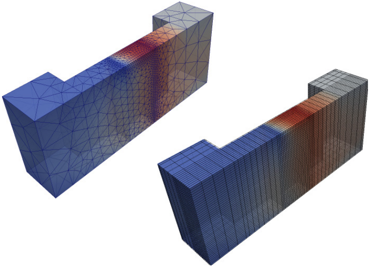
Other Discretizations

Finite Volumes

- Iterations over vertices: Similar to finite differences
- Beware of advanced schemes (depends)

Finite Elements

- Iteration over elements (cells)
- BUT: write matrix entries associated with vertices
- $\Rightarrow$ Avoid race conditions
- More preparation required before we can address this



Next Step

Sequential to Parallel: ILU Example

(in preparation for Finite Element assembly strategies)

Parallel ILU

ILU - Basic Idea

- Factor sparse matrix $\boldsymbol{A} \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$
- $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ sparse, triangular
- ILU0: Pattern of $\tilde{\boldsymbol{L}}$, $\tilde{\boldsymbol{U}}$ equal to $\boldsymbol{A}$
- ILUT: Keep k elements per row

Solver Cycle Phase

- Residual correction $\color{red}{\tilde{\boldsymbol{L}}}\color{black}\tilde{\boldsymbol{U}}x = z$
- Forward solve $\color{red}{\tilde{\boldsymbol{L}}}\color{black}y = z$
- Backward solve $\tilde{\boldsymbol{U}}x = y$
- Little parallelism in general

$$\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\color{red}\times & 3 & \times & & & & & & \\
\color{red}\times & \color{red}\times & 4 & \times & & & & & \\
\color{red}\times & & \color{red}\times & 5 & \times & \times & & & \times \\
 & & & \color{red}\times & 5 & \times & & \times & \times \\
\color{red}\times & & & \color{red}\times & \color{red}\times & 6 & \times & \times & \\
\color{red}\times & & & & & \color{red}\times & 3 & & \\
 & & & & \color{red}\times & \color{red}\times & & 4 & \times \\
 & & & \color{red}\times & \color{red}\times & & & \color{red}\times & 4
\end{pmatrix}$$

Parallel ILU

ILU Level Scheduling

- Build dependency graph
- Substitute as many entries as possible simultaneously
- Trade-off: Each step vs. multiple steps in a single kernel

$$
\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\times & 3 & \times & & & & & & \\
\times & \times & 4 & \times & & & & & \\
\times & & \times & 5 & \times & \times & & & \times \\
 & & & \times & 5 & \times & & \times & \times \\
\times & & & \times & \times & 6 & \times & \times & \\
\times & & & & & \times & 3 & & \\
 & & & & \times & \times & & 4 & \times \\
 & & & \times & \times & & & \times & 4
\end{pmatrix}
$$

Parallel ILU

ILU Level Scheduling

- Build dependency graph
- Substitute as many entries as possible simultaneously
- Trade-off: Each step vs. multiple steps in a single kernel

$$
\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\times & 3 & \times & & & & & & \\
\times & \times & 4 & \times & & & & & \\
\times & & \times & 5 & \times & \times & & & \times \\
 & & & \times & 5 & \times & & \times & \times \\
\times & & & \times & \times & 6 & \times & \times & \\
\times & & & & & \times & 3 & & \\
 & & & & \times & \times & & 4 & \times \\
 & & & \times & \times & & & \times & 4
\end{pmatrix}
$$

Parallel ILU

ILU Level Scheduling

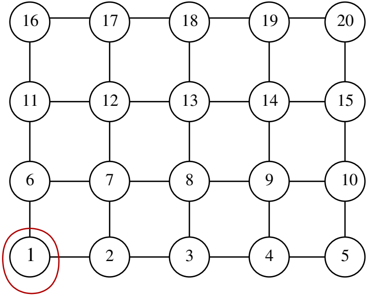
- Build dependency graph
- Substitute as many entries as possible simultaneously
- Trade-off: Each step vs. multiple steps in a single kernel

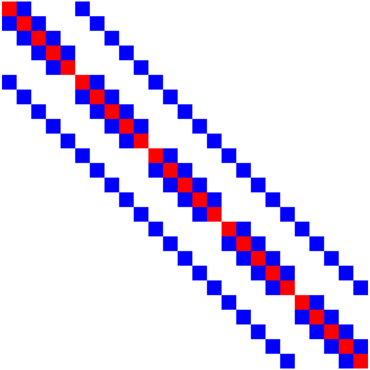
$$
\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\times & 3 & \times & & & & & & \\
\times & \times & 4 & \times & & & & & \\
\times & & \times & 5 & \times & \times & & & \times \\
 & & & \times & 5 & \times & & \times & \times \\
\times & & & \times & \times & 6 & \times & \times & \\
\times & & & & & \times & 3 & & \\
 & & & & \times & \times & & 4 & \times \\
 & & & \times & \times & & & \times & 4
\end{pmatrix}
$$

Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

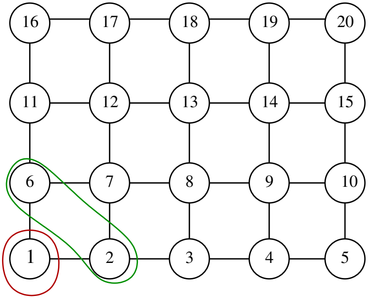


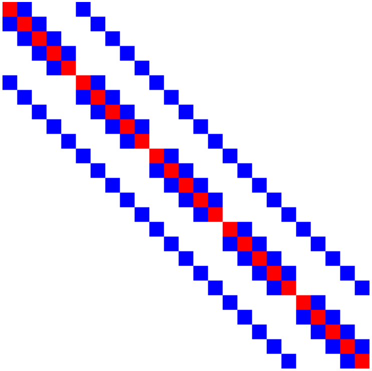


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

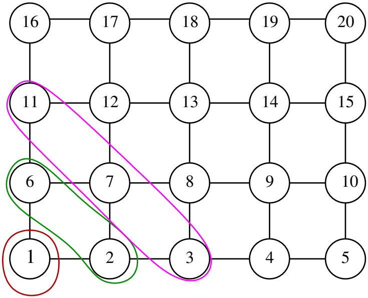


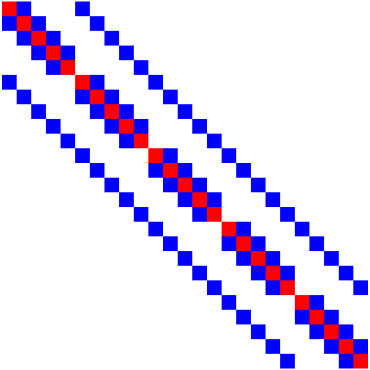


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

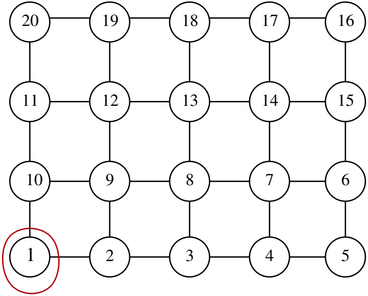


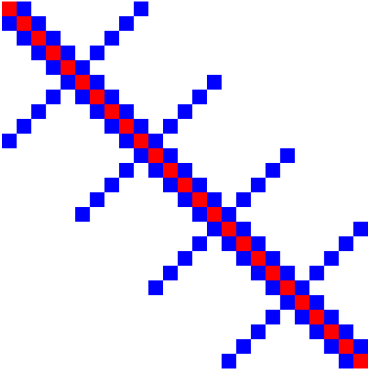


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

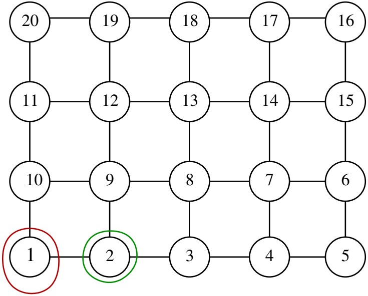


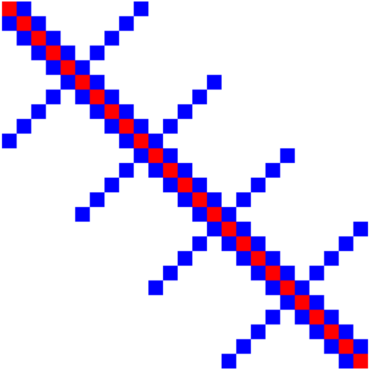


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

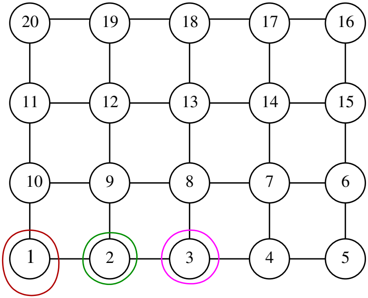


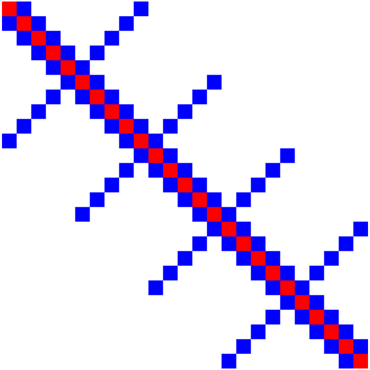


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

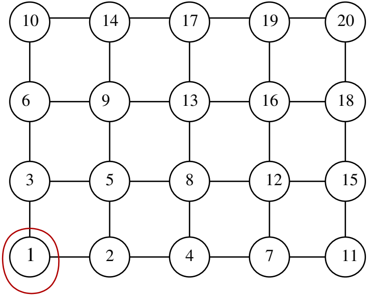


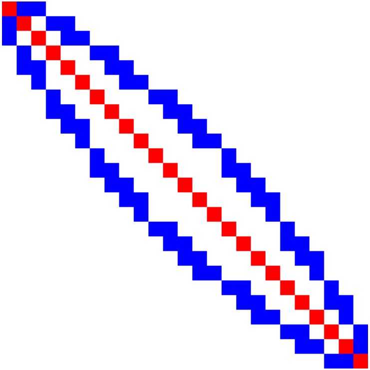


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

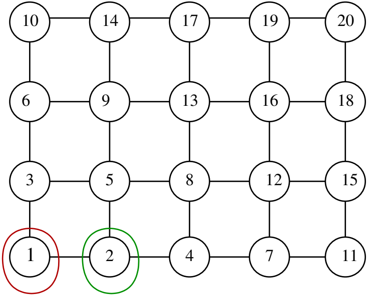


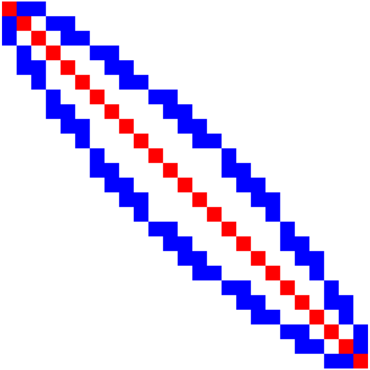


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

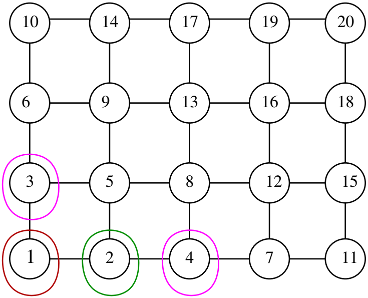


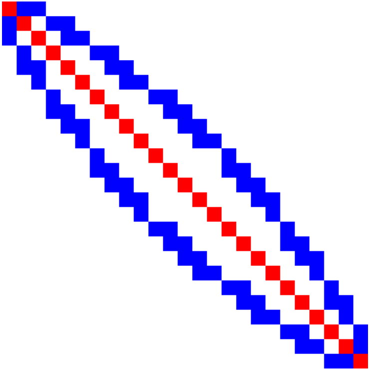


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

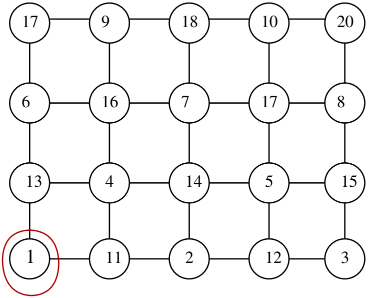


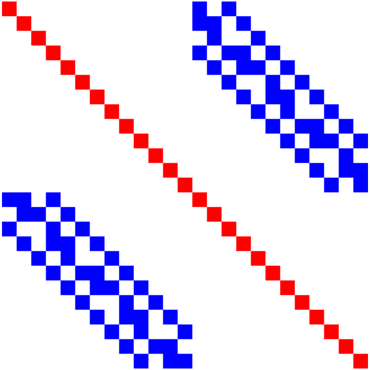


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

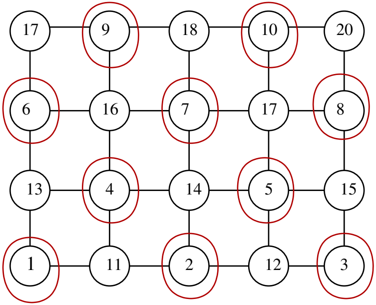


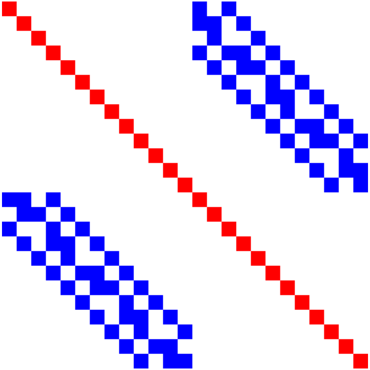


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

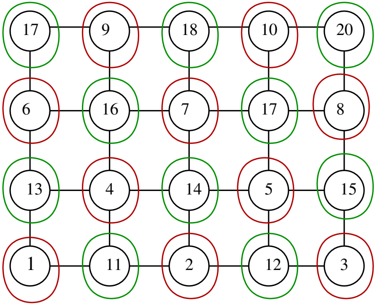


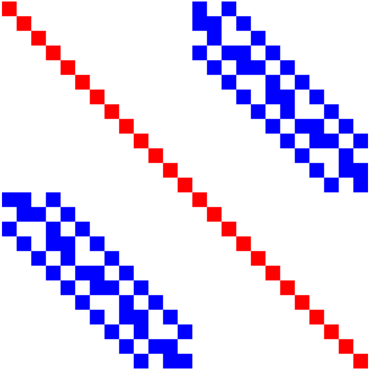


Parallel ILU

ILU Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d





Parallel ILU - Innovation

Sequential

```
for i=2..n
  for k=1..i-1, (i,k)in A
```

$$a_{ik} = a_{ik}/a_{kk}$$

```
    for j=k+1..n, (i,j)in A
```

$$u_{ij} = a_{ij} - a_{ik}a_{kj}$$

Parallel

```
for (sweep = 1, 2, ...)
 parallel for (i,j)in A
   if (i > j)
```

$$l_{ij} = (a_{ij} - \sum_{k=1}^{j=1} l_{ik}u_{kj})/u_{jj}$$

```
   else
```

$$u_{ij} = a_{ij} - \sum_{k=1}^{=1} l_{ik}u_{kj}$$

Fine-Grained Parallel ILU Setup

- Proposed by Chow and Patel (SISC, vol. 37(2)) for CPUs and MICs
- Massively parallel (one thread per row)

Parallel ILU - Innovation

Sequential

```
for i=2..n
  for k=1..i-1, (i,k)in A
```

$$a_{ik} = a_{ik}/a_{kk}$$

```
    for j=k+1..n, (i,j)in A
```

$$u_{ij} = a_{ij} - a_{ik}a_{kj}$$

Parallel

```
for (sweep = 1, 2, ...)
  parallel for (i,j)in A
    if (i > j)
```

$$l_{ij} = (a_{ij} - \sum_{k=1}^{j=1} l_{ik}u_{kj})/u_{jj}$$

```
    else
```

$$u_{ij} = a_{ij} - \sum_{k=1}^{=1} l_{ik}u_{kj}$$

Fine-Grained Parallel ILU Setup

- Proposed by Chow and Patel (SISC, vol. 37(2)) for CPUs and MICs
- Massively parallel (one thread per row)

Preconditioner Application

- Truncated Neumann series:

$$\mathbf{L}^{-1} \approx \sum_{k=0}^{K} (\mathbf{I} - \mathbf{L})^k, \quad \mathbf{U}^{-1} \approx \sum_{k=0}^{K} (\mathbf{I} - \mathbf{U})^k$$

- Exact triangular solves not necessary

Exercises

Environment

- `https://gtx1080.360252.org/2020/ex5/`
- (Might receive visual updates and additional hints over the next days)
- Due: Tuesday, November 24, 2020 at 23:59pm

Hints and Suggestions

- Consider version control for locally developed code
- Please let me know of any bugs or issues