Query Processing on Dynamic Networks with Customizable Contraction Hierarchies on Neo4j

Marius Hahn

Grouf of Database and Information Systems
Department of Computer and Information Science
Faculty of Sciences
Universität Konstanz

Master Colloquium, 14th December 2023

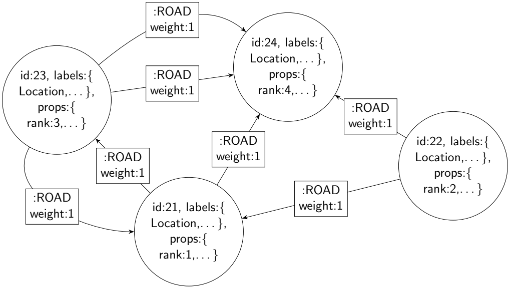
Table of Contents

Motivation and Context

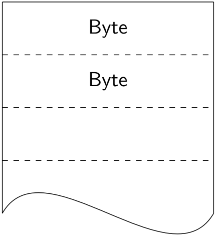
- Context ⇛ Graph Databases
- External memory
- Accelerate Shortest Path Queries in Databases
- Why Customizable Contraction Hierarchies?
  - fast for main memory applications
  - reasonable preprocessing time
  - It is updatable
- Test Data ⇛ Road Networks

Obstacles

Property Graph:



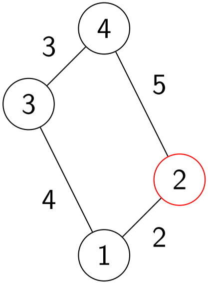
File:



- Databases use HDDs ⇛ slow random access
- transformation to one dimensional data structure

Table of Contents

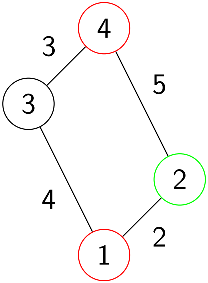
Let's go from v_2 to v_3



id	dist	settled
2	0	false

Dijkstra

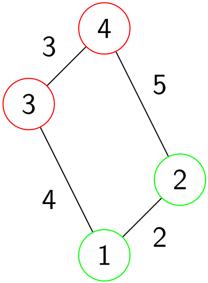
Let's go from v_2 to v_3



id	dist	settled
2	0	true
1	2	false
4	5	false

Dijkstra

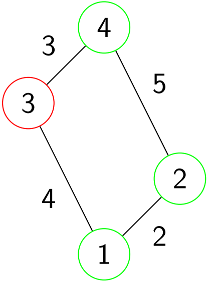
Let's go from v_2 to v_3



id	dist	settled
2	0	true
1	2	true
4	5	false
3	6	false

Dijkstra

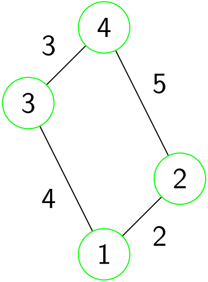
Let's go from v_2 to v_3



id	dist	settled
2	0	true
1	2	true
4	5	true
3	6	false

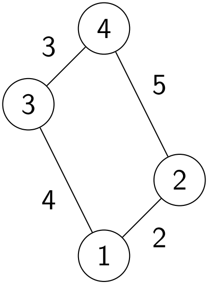
Dijkstra

Let's go from v_2 to v_3

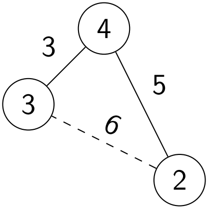


id	dist	settled
2	0	true
1	2	true
4	5	true
3	6	true

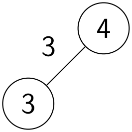
Contraction Hierarchies Example



Contraction Hierarchies Example



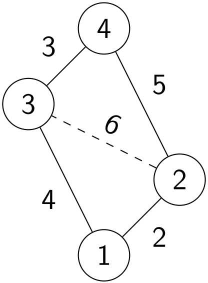
Contraction Hierarchies Example



Contraction Hierarchies Example

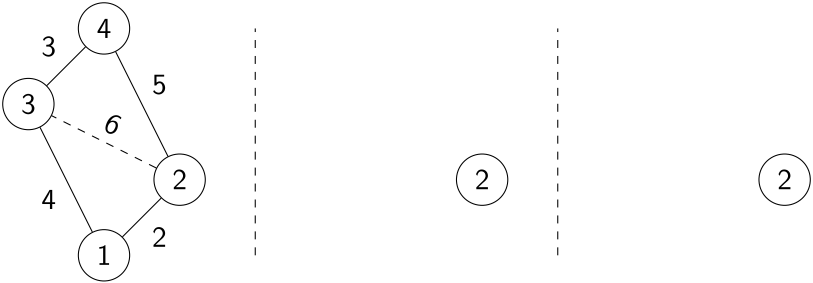
Contraction Hierarchies Example

Let's go from v_2 to v_3



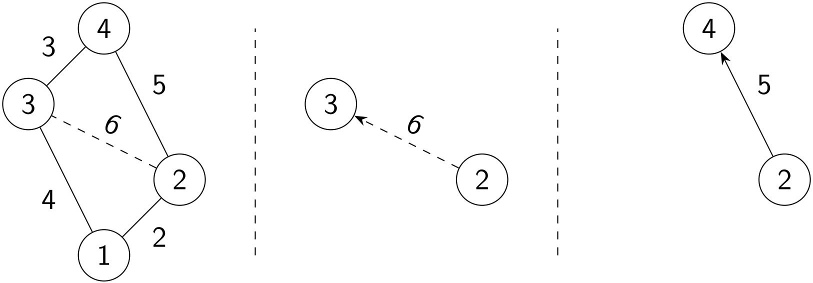
Contraction Hierarchies Example

Let's go from v_2 to v_3



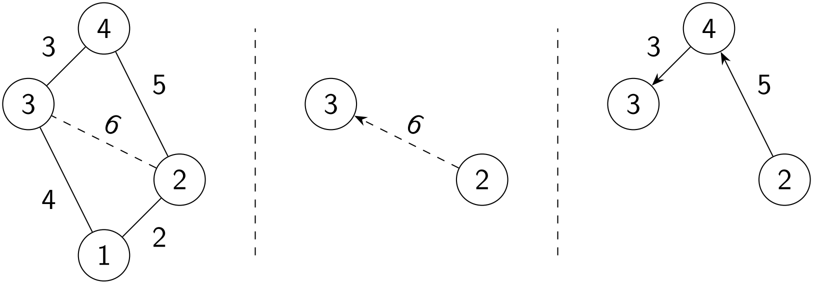
Contraction Hierarchies Example

Let's go from v_2 to v_3

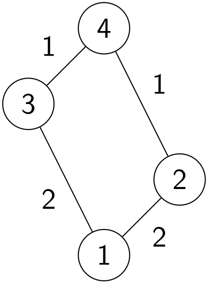


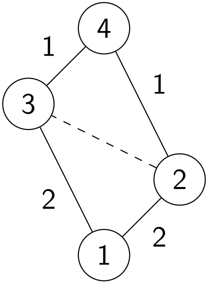
Contraction Hierarchies Example

Let's go from v_2 to v_3



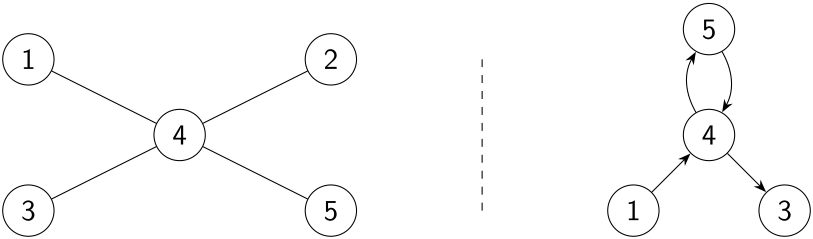
- CH insert shortcut if shortest path property is violated
- CCH insert shortcut if there is no direct connection





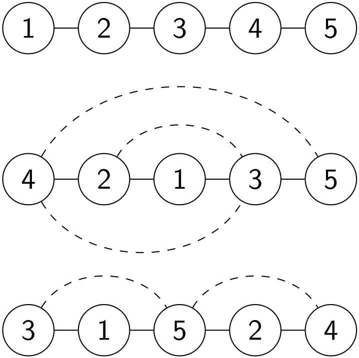
Important vertex not contraced Last

- Contracted Using Edge Difference
- Go from v(1) to v(3)
- Forward and Backward search are deeper that the should be
- Switch contraction order of v(4) and v(5)

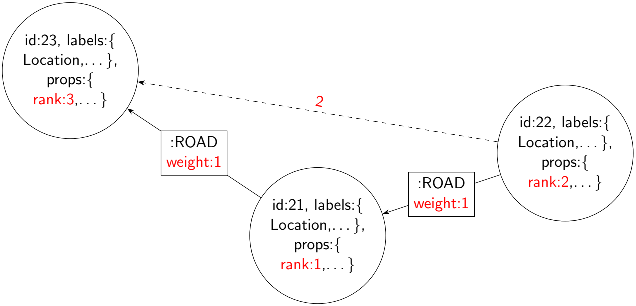


Linear Contraction

1. linear contraction
    - No Shortcuts
    - Could happen with ED
    - four vertices to expand
2. middle vertex first
    - Three Shortcuts
    - four vertices to expand
3. good contraction order
    - Three Shortcuts
    - four vertices to expand

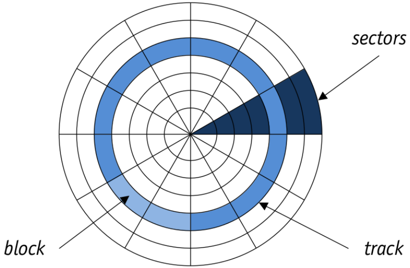


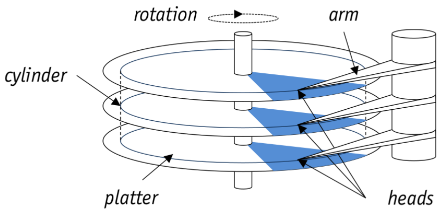
Persistance Objectives



- keep only necessary data
  - rank → to do the mapping to the input graph
  - arc weight
- Store edges that are likely to be request together spacial close
- Use as few space as possible → the less you write the less you read

Magnetic Disks





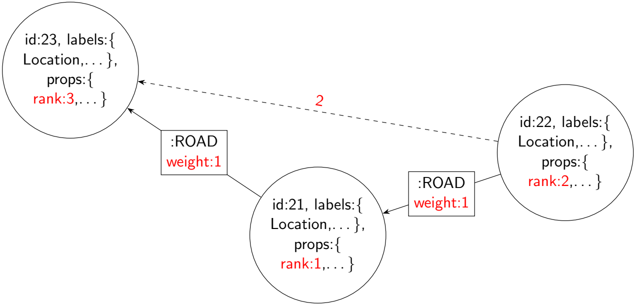
- Data is arranged in concentric rings (tracks) on platters
- Tracks are divided into arc-shaped sectors

Important theorem

Data is read from and written to disk one **block** at a time

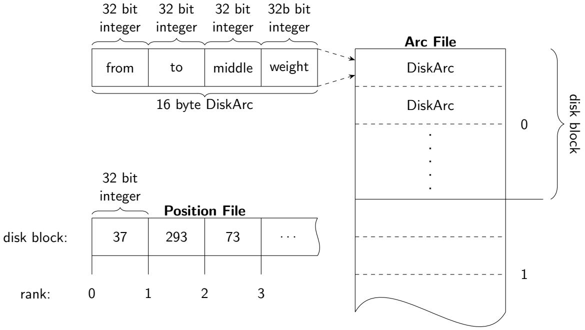
Transformation to a Table

- Depth-First-Search starting at highest rank
- retrieve only arcs. vertices will be reconstructed form arcs
- remember middle node



start rank	end rank	middle rank	weight
2	3	1	2
1	3	-1	1
2	1	-1	1

Store Example



- fill all arcs of a vertex into a block
- add block number of rank to position file. $G_\uparrow$ use *from* ; $G_\downarrow$ use *to*
- if next vertex' arcs don't fit anymore → flush block and take next

Min Block Size

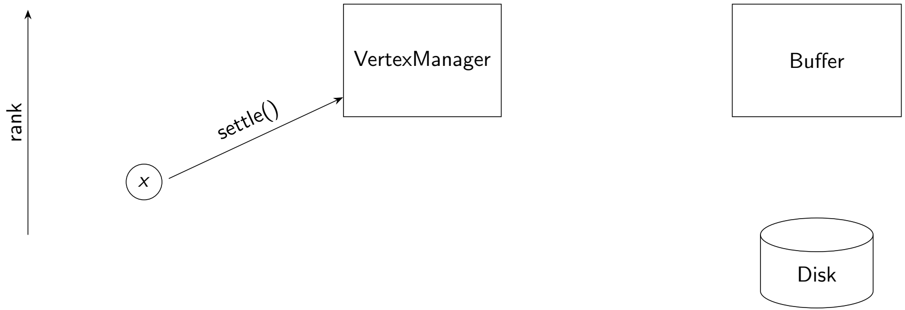
$$max(d_{\uparrow max}(v), d_{\downarrow max}(v)) \leqslant \frac{diskBlockSize}{16}$$

CCH Disk Search (upwards graph example)

1. lazy load vertices ⇒ only start node is loaded without arcs

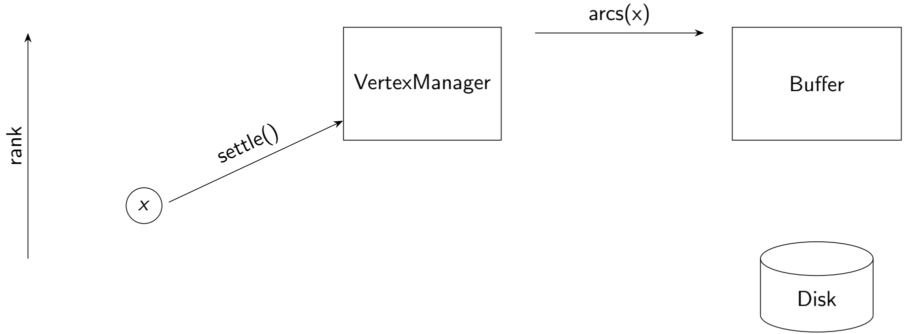
CCH Disk Search (upwards graph example)

1. lazy load vertices ⇒ only start node is loaded without arcs
2. settle vertex (right before expanding it)



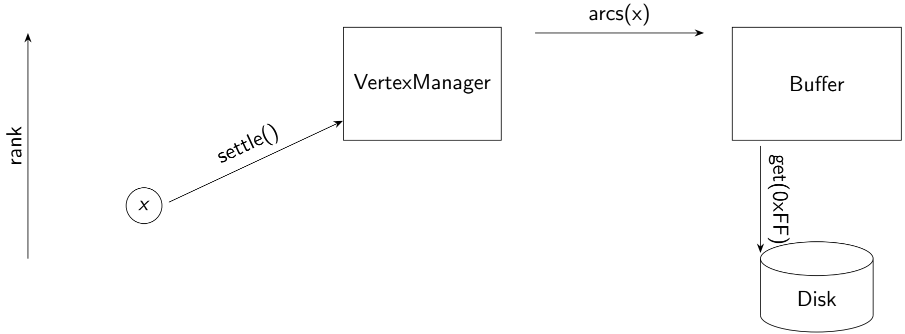
CCH Disk Search (upwards graph example)

1. lazy load vertices $\Rrightarrow$ only start node is loaded without arcs
2. settle vertex (right before expanding it)
   - VertexManager requests arc of v(x) from buffer



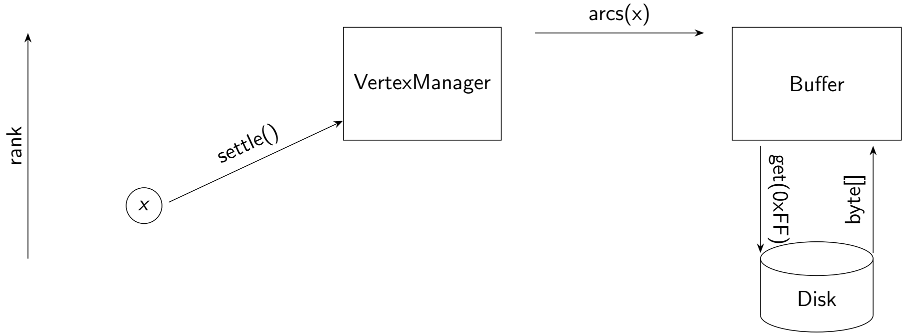
CCH Disk Search (upwards graph example)

1. lazy load vertices ⇛ only start node is loaded without arcs
2. settle vertex (right before expanding it)
   - VertexManager requests arc of v(x) from buffer
   - Buffer requests arcs from disk if not cached yet



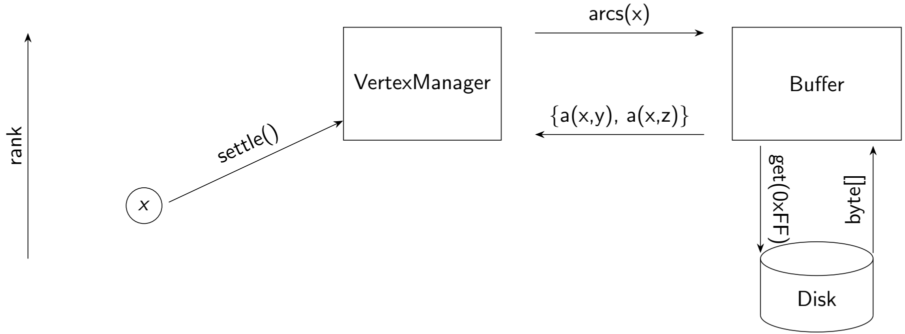
CCH Disk Search (upwards graph example)

1. lazy load vertices $\Rrightarrow$ only start node is loaded without arcs
2. settle vertex (right before expanding it)
   - VertexManager requests arc of v(x) from buffer
   - Buffer requests arcs from disk if not cached yet



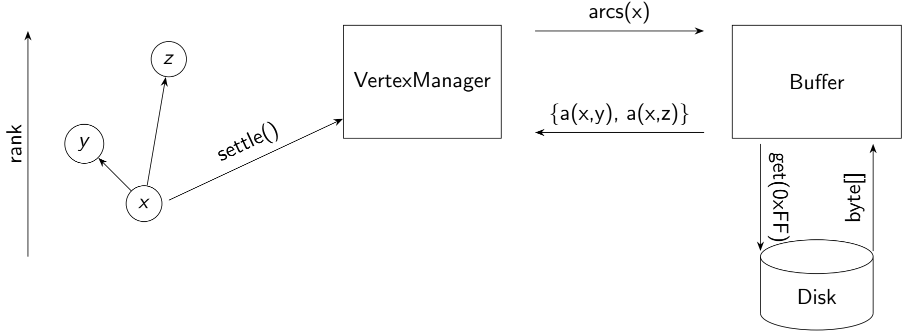
CCH Disk Search (upwards graph example)

1. lazy load vertices $\Rrightarrow$ only start node is loaded without arcs
2. settle vertex (right before expanding it)
   - VertexManager requests arc of v(x) from buffer
   - Buffer requests arcs from disk if not cached yet
   - Buffer returns arcs

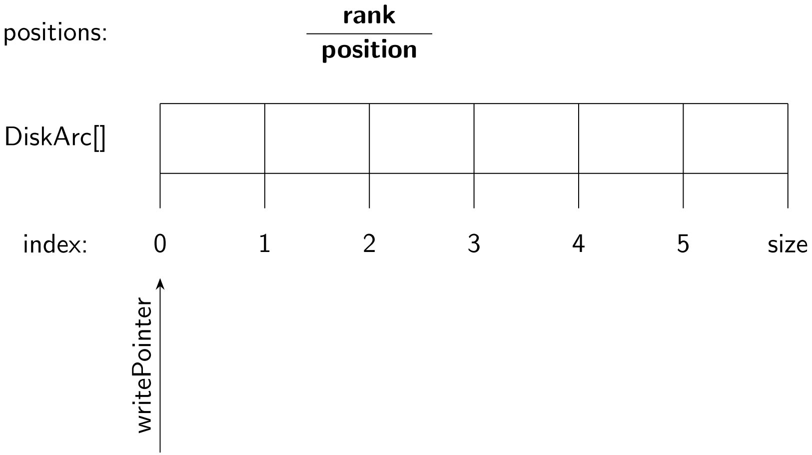


CCH Disk Search (upwards graph example)

1. lazy load vertices ⇛ only start node is loaded without arcs
2. settle vertex (right before expanding it)
   - VertexManager requests arc of v(x) from buffer
   - Buffer requests arcs from disk if not cached yet
   - Buffer returns arcs
   - VertexManager attaches arcs to v(x)



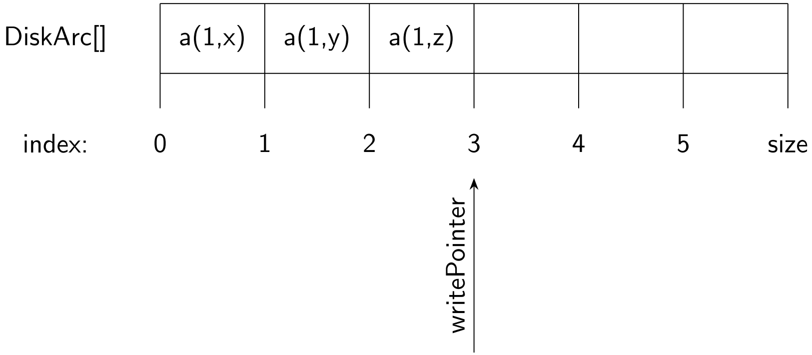
Circular Buffer



Circular Buffer

positions:

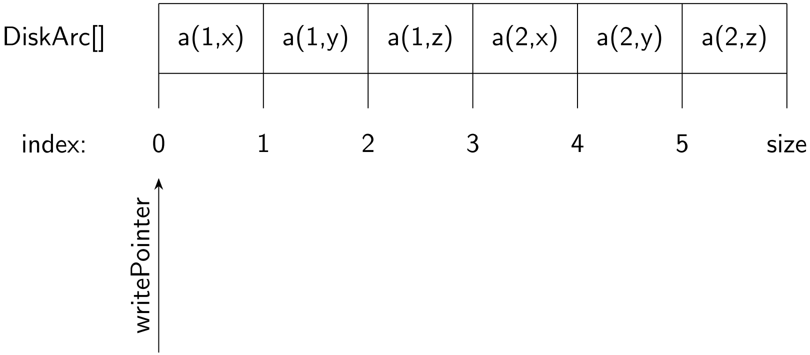
rank	1
position	2



Circular Buffer

positions:

rank	1	2
position	2	5



Circular Buffer

positions:

rank	1	2	max(rank)
position	2	5	-1

DiskArc[]

a(1,x)	a(1,y)	a(1,z)	a(2,x)	a(2,y)	a(2,z)

index: 0 1 2 3 4 5 size

writePointer

Circular Buffer

positions:

rank	1	2	3	max(rank)
position	2	5	3	-1

DiskArc[]

a(3,x)	a(3,y)	a(3,z)	a(3,zx)	a(2,y)	a(2,z)

index: 0 1 2 3 4 5 size

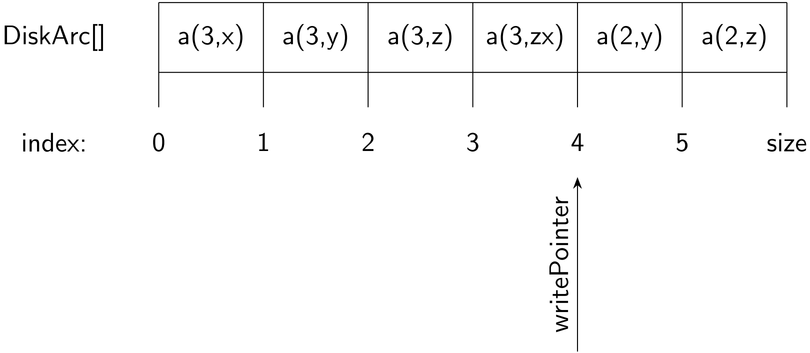
writePointer

remove incomplete edge set from position

Circular Buffer

positions:

rank	1	3	max(rank)
position	2	3	-1



Circular Buffer

positions:

rank	1	3	max(rank)
position	2	3	-1

DiskArc[]

a(3,x)	a(3,y)	a(3,z)	a(3,zx)	a(2,y)	a(2,z)

index: 0 1 2 3 4 5 size

- Retrieve Arcs $\Rrightarrow$ iterate backwards from position until start vertex differs
- If arc is doesn't start with requested rank $\Rrightarrow$ remove position and refetch

In this slide

In this slide
the text will be partially visible

In this slide
the text will be partially visible
And finally everything will be there

Table of Contents

Sample frame title

In this slide, some important text will be highlighted because it's important. Please, don't abuse it.

Remark

Sample text

Important theorem

Sample text in red box

Examples

Sample text in green box. The title of the block is "Examples".

Two-column slide

This is a text in first column.

$$E = mc^2$$

- First item
- Second item

This text will be in the second column and on a second tought this is a nice looking layout in some cases.