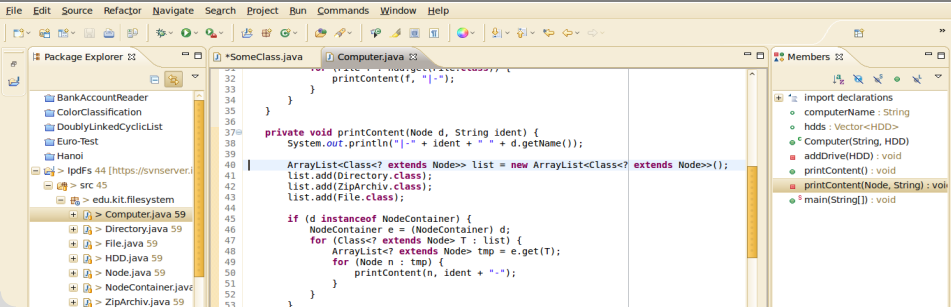


# How Google searches work

History, Algorithm

Martin Thoma, Benjamin | 24. Januar 2013

FAKULTÄT FÜR INFORMATIK



The screenshot shows an IDE with the following components:

- Package Explorer:** Shows a project structure with packages like `edu.kit.filesystem` and classes like `Computer.java`, `Directory.java`, `File.java`, `HDD.java`, `Node.java`, `NodeContainer.java`, and `ZipArchiv.java`.
- Editor:** Displays the code for `Computer.java`. The code includes a `printContent` method that prints the content of a `Node` and its children. It also shows a `main` method that initializes a `NodeContainer` and prints its content.
- Members:** Shows the members of the `Computer` class, including `computerName`, `hdds`, `addDrive`, `printContent`, and `main`.

```
32     printContent(f, "-");
33   }
34 }
35
36
37 private void printContent(Node d, String ident) {
38     System.out.println(ident + " " + d.getName());
39
40     ArrayList<Class? extends Node> list = new ArrayList<Class? extends Node>();
41     list.add(Directory.class);
42     list.add(ZipArchiv.class);
43     list.add(File.class);
44
45     if (d instanceof NodeContainer) {
46         NodeContainer e = (NodeContainer) d;
47         for (Class<? extends Node> T : list) {
48             ArrayList<? extends Node> tmp = e.get(T);
49             for (Node n : tmp) {
50                 printContent(n, ident + "-");
51             }
52         }
53     }
```

1 Introduction

2 PageRank

3 End

# The early days

In the beginning, there were only web catalogues (maintained by hand)

# The early days

Web crawler

- Humans know what is good for them
  - Humans create Websites
  - Humans will only [link](#) to Websites they like
- ⇒ Hyperlinks are a quality indicator

- Humans know what is good for them
  - Humans create Websites
  - Humans will only [link](#) to Websites they like
- ⇒ Hyperlinks are a quality indicator

- Humans know what is good for them
- Humans create Websites
- Humans will only [link](#) to Websites they like

⇒ Hyperlinks are a quality indicator

- Humans know what is good for them
  - Humans create Websites
  - Humans will only [link](#) to Websites they like
- ⇒ Hyperlinks are a quality indicator



# How could we use that?

- Simply count number of links to a Website
- ✗ 10,000 links from only one page
- Count numbers of Websites that link to a Website
- ✗ Quality of the page matters
- ✗ Total number of links on the source page matters

# How could we use that?

- Simply count number of links to a Website
- ✗ 10,000 links from only one page
- Count numbers of Websites that link to a Website
- ✗ Quality of the page matters
- ✗ Total number of links on the source page matters

# How could we use that?

- Simply count number of links to a Website
- ✗ 10,000 links from only one page
- Count numbers of Websites that link to a Website
- ✗ Quality of the page matters
- ✗ Total number of links on the source page matters

# How could we use that?

- Simply count number of links to a Website
- ✗ 10,000 links from only one page
- Count numbers of Websites that link to a Website
- ✗ Quality of the page matters
- ✗ Total number of links on the source page matters

# How could we use that?

- Simply count number of links to a Website
- ✗ 10,000 links from only one page
- Count numbers of Websites that link to a Website
- ✗ Quality of the page matters
- ✗ Total number of links on the source page matters

# A brilliant idea



Sergey Brin



Larry Page

- Decisions of humans are complicated

- A lot of webpages get visited

⇒ modellize clicks on links as random behaviour

- Links are important

- Links of page A get less important, if A has many links

- Links of page A get more important, if many link to A

⇒ if B has a link from A, the rank of B increases by  $\frac{Rank(A)}{Links(A)}$

**if A links to B then**

$$Rank(B) += \frac{Rank(A)}{Links(A)}$$

- Decisions of humans are complicated
- A lot of webpages get visited

⇒ modelize clicks on links as random behaviour

- Links are important
- Links of page A get less important, if A has many links
- Links of page A get more important, if many link to A

⇒ if B has a link from A, the rank of B increases by  $\frac{Rank(A)}{Links(A)}$

**if A links to B then**

$$Rank(B) += \frac{Rank(A)}{Links(A)}$$



- Decisions of humans are complicated

- A lot of webpages get visited

⇒ modellize clicks on links as random behaviour

- Links are important

- Links of page A get less important, if A has many links

- Links of page A get more important, if many link to A

⇒ if B has a link from A, the rank of B increases by  $\frac{Rank(A)}{Links(A)}$

if A links to B then

$$Rank(B) += \frac{Rank(A)}{Links(A)}$$

- Decisions of humans are complicated

- A lot of webpages get visited

⇒ modellize clicks on links as random behaviour

- Links are important

- Links of page A get less important, if A has many links

- Links of page A get more important, if many link to A

⇒ if B has a link from A, the rank of B increases by  $\frac{Rank(A)}{Links(A)}$

if A links to B then

$$Rank(B) += \frac{Rank(A)}{Links(A)}$$

- Decisions of humans are complicated

- A lot of webpages get visited

⇒ modellize clicks on links as random behaviour

- Links are important

- Links of page A get less important, if A has many links

- Links of page A get more important, if many link to A

⇒ if B has a link from A, the rank of B increases by  $\frac{Rank(A)}{Links(A)}$

if A links to B then

$$Rank(B) += \frac{Rank(A)}{Links(A)}$$

- Decisions of humans are complicated

- A lot of webpages get visited

⇒ modellize clicks on links as random behaviour

- Links are important

- Links of page A get less important, if A has many links

- Links of page A get more important, if many link to A

⇒ if B has a link from A, the rank of B increases by  $\frac{Rank(A)}{Links(A)}$

if A links to B then

$$Rank(B) += \frac{Rank(A)}{Links(A)}$$

- Decisions of humans are complicated

- A lot of webpages get visited

⇒ modellize clicks on links as random behaviour

- Links are important

- Links of page A get less important, if A has many links

- Links of page A get more important, if many link to A

⇒ if B has a link from A, the rank of B increases by  $\frac{Rank(A)}{Links(A)}$

if A links to B then

$$Rank(B) += \frac{Rank(A)}{Links(A)}$$

- Decisions of humans are complicated

- A lot of webpages get visited

⇒ modellize clicks on links as random behaviour

- Links are important

- Links of page A get less important, if A has many links

- Links of page A get more important, if many link to A

⇒ if B has a link from A, the rank of B increases by  $\frac{Rank(A)}{Links(A)}$

**if A links to B then**

$$Rank(B) += \frac{Rank(A)}{Links(A)}$$

- Websites = nodes = anthill
- Links = edges = paths
- You place ants on each node
- They walk over the paths  
(at random, they are ants!)
- After some time, some anthills will have more ants than others
- Those hills are more attractive than others
- # ants is probability that a random user would end on a website

- Websites = nodes = anthill
- Links = edges = paths
- You place ants on each node
- They walk over the paths  
(at random, they are ants!)
- After some time, some anthills will have more ants than others
- Those hills are more attractive than others
- # ants is probability that a random user would end on a website



- Websites = nodes = anthill
- Links = edges = paths
- You place ants on each node
- They walk over the paths  
(at random, they are ants!)
- After some time, some anthills will have more ants than others
- Those hills are more attractive than others
- # ants is probability that a random user would end on a website

- Websites = nodes = anthill
- Links = edges = paths
- You place ants on each node
- They walk over the paths  
(at random, they are ants!)
- After some time, some anthills will have more ants than others
- Those hills are more attractive than others
- # ants is probability that a random user would end on a website

- Websites = nodes = anthill
- Links = edges = paths
- You place ants on each node
- They walk over the paths  
(at random, they are ants!)
- After some time, some anthills will have more ants than others
- Those hills are more attractive than others
- # ants is probability that a random user would end on a website

- Websites = nodes = anthill
- Links = edges = paths
- You place ants on each node
- They walk over the paths  
(at random, they are ants!)
- After some time, some anthills will have more ants than others
- Those hills are more attractive than others
- # ants is probability that a random user would end on a website

- Websites = nodes = anthill
- Links = edges = paths
- You place ants on each node
- They walk over the paths  
(at random, they are ants!)
- After some time, some anthills will have more ants than others
- Those hills are more attractive than others
- # ants is probability that a random user would end on a website

- Websites = nodes = anthill
- Links = edges = paths
- You place ants on each node
- They walk over the paths  
(at random, they are ants!)
- After some time, some anthills will have more ants than others
- Those hills are more attractive than others
- # ants is probability that a random user would end on a website

Let  $x$  be a web page. Then

- $L(x)$  is the set of Websites that link to  $x$
- $C(y)$  is the out-degree of page  $y$
- $\alpha$  is probability of random jump
- $N$  is the total number of websites

$$PR(x) := \alpha \left( \frac{1}{N} \right) + (1 - \alpha) \sum_{y \in L(x)} \frac{PR(y)}{C_y}$$

```
function PAGERANK(Graph web, double  $q = 0.15$ , int iterations)  
  for all  $page \in G$  do  
     $page.pageRank = \frac{1}{|G|}$  ▷ initial probability  
  end for  
  while  $iterations > 0$  do  
    for all  $page \in G$  do ▷ calculate pageRank of page  
       $page.pageRank = q$   
      for all  $y \in L(page)$  do  
         $page.pageRank += \frac{y.pageRank}{C(y)}$   
      end for  
    end for  
     $iterations -= 1$ 
```



```
function PAGERANK(Graph web, double  $q = 0.15$ , int iterations)  
  for all page  $\in G$  do  
    page.pageRank =  $\frac{1}{|G|}$  ▷ initial probability  
  end for  
  while iterations > 0 do  
    for all page  $\in G$  do ▷ calculate pageRank of page  
      page.pageRank =  $q$   
      for all  $y \in L(\textit{page})$  do  
        page.pageRank +=  $\frac{y.\textit{pageRank}}{C(y)}$   
      end for  
    end for  
    iterations -= 1
```

```
function PAGERANK(Graph web, double  $q = 0.15$ , int iterations)  
  for all  $page \in G$  do  
     $page.pageRank = \frac{1}{|G|}$  ▷ initial probability  
  end for  
  while  $iterations > 0$  do  
    for all  $page \in G$  do ▷ calculate pageRank of page  
       $page.pageRank = q$   
      for all  $y \in L(page)$  do  
         $page.pageRank += \frac{y.pageRank}{C(y)}$   
      end for  
    end for  
     $iterations -= 1$ 
```

```
function PAGERANK(Graph web, double  $q = 0.15$ , int iterations)  
  for all  $page \in G$  do  
     $page.pageRank = \frac{1}{|G|}$  ▷ initial probability  
  end for  
  while  $iterations > 0$  do  
    for all  $page \in G$  do ▷ calculate pageRank of page  
       $page.pageRank = q$   
      for all  $y \in L(page)$  do  
         $page.pageRank += \frac{y.pageRank}{C(y)}$   
      end for  
    end for  
     $iterations -= 1$ 
```

```
function PAGERANK(Graph web, double  $q = 0.15$ , int iterations)  
  for all  $page \in G$  do  
     $page.pageRank = \frac{1}{|G|}$  ▷ initial probability  
  end for  
  while  $iterations > 0$  do  
    for all  $page \in G$  do ▷ calculate pageRank of page  
       $page.pageRank = q$   
      for all  $y \in L(page)$  do  
         $page.pageRank += \frac{y.pageRank}{C(y)}$   
      end for  
    end for  
     $iterations -= 1$ 
```

```
function PAGERANK(Graph web, double  $q = 0.15$ , int iterations)  
  for all  $page \in G$  do  
     $page.pageRank = \frac{1}{|G|}$  ▷ initial probability  
  end for  
  while  $iterations > 0$  do  
    for all  $page \in G$  do ▷ calculate pageRank of page  
       $page.pageRank = q$   
      for all  $y \in L(page)$  do  
         $page.pageRank += \frac{y.pageRank}{C(y)}$   
      end for  
    end for  
     $iterations -= 1$ 
```

```
function PAGERANK(Graph web, double  $q = 0.15$ , int iterations)  
  for all  $page \in G$  do  
     $page.pageRank = \frac{1}{|G|}$  ▷ initial probability  
  end for  
  while  $iterations > 0$  do  
    for all  $page \in G$  do ▷ calculate pageRank of page  
       $page.pageRank = q$   
      for all  $y \in L(page)$  do  
         $page.pageRank += \frac{y.pageRank}{C(y)}$   
      end for  
    end for  
     $iterations -= 1$ 
```

```
function PAGERANK(Graph web, double  $q = 0.15$ , int iterations)  
  for all  $page \in G$  do  
     $page.pageRank = \frac{1}{|G|}$  ▷ initial probability  
  end for  
  while  $iterations > 0$  do  
    for all  $page \in G$  do ▷ calculate pageRank of page  
       $page.pageRank = q$   
      for all  $y \in L(page)$  do  
         $page.pageRank \mathrel{+}= \frac{y.pageRank}{C(y)}$   
      end for  
    end for  
     $iterations -= 1$ 
```

# What you've learned

- Web catalogues
- Webcrawler
- Graph (nodes, edges)
- Random walk (ants)
- PageRank
- Read Pseudocode



- [Sergey Brin](#) by enlewof
- [Larry Page](#) by aweigend

# Thank you for your attention!

Days 1 - 10

Teach yourself variables, constants, arrays, strings, expressions, statements, functions,...



Days 11 - 21

Teach yourself program flow, pointers, references, classes, objects, inheritance, polymorphism, ...



Days 22 - 697

Do a lot of recreational programming. Have fun hacking but remember to learn from your mistakes.



Days 698 - 3648

Interact with other programmers. Work on programming projects together. Learn from them.



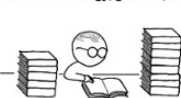
Days 3649 - 7781

Teach yourself advanced theoretical physics and formulate a consistent theory of quantum gravity.



Days 7782 - 14611

Teach yourself biochemistry, molecular biology, genetics,...



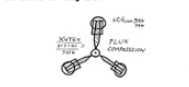
Day 14611

Use knowledge of biology to make an age-reversing potion.



Day 14611

Use knowledge of physics to build flux capacitor and go back in time to day 21.



Day 21

Replace younger self.



As far as I know, this is the easiest way to "Teach Yourself C++ in 21 Days".