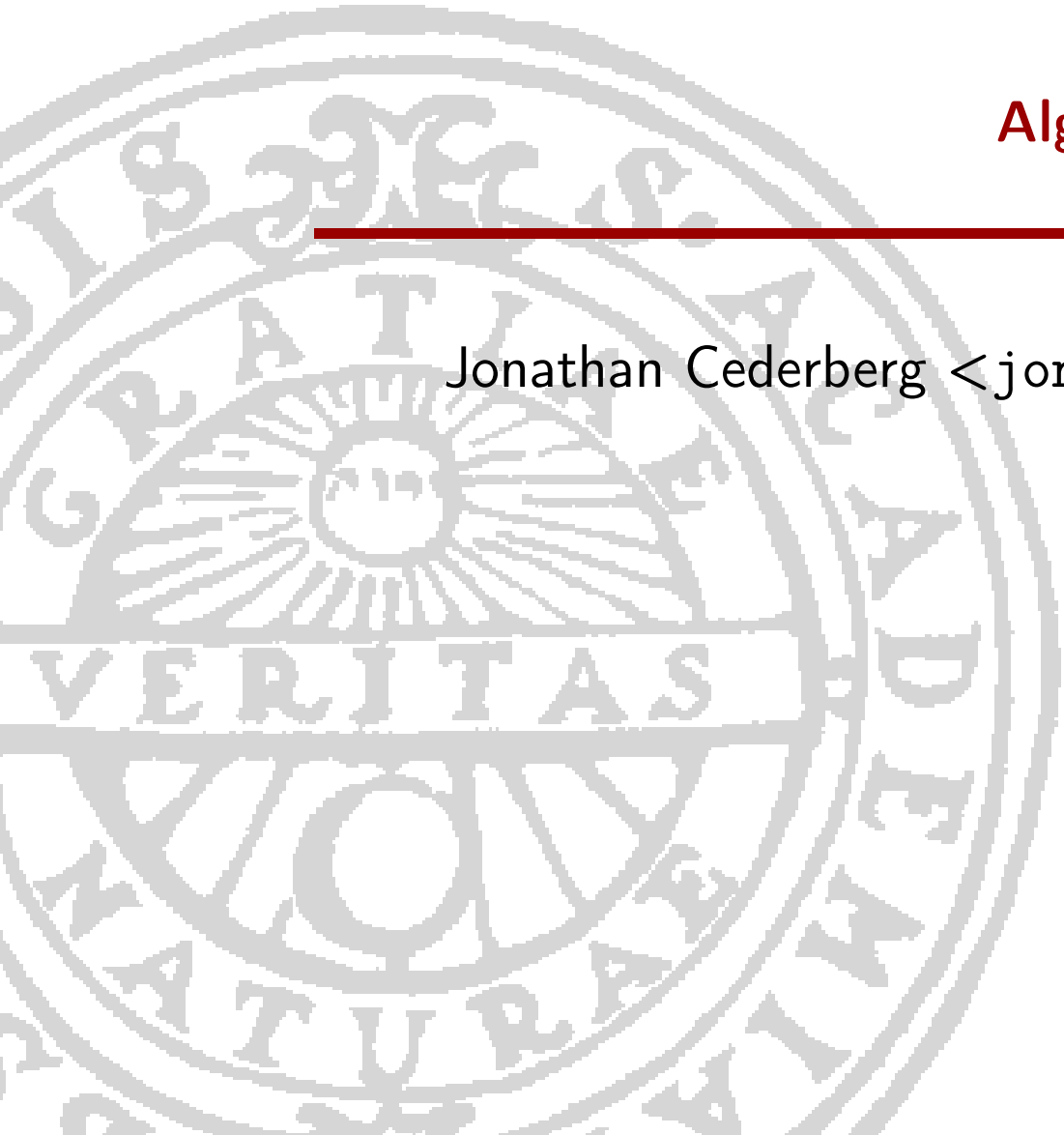


Tutorial 4

Algorithms and Data Structures

Jonathan Cederberg <jonathan.cederberg@it.uu.se>

Friday, October 7^{th}, 2011

UPPSALA
UNIVERSITET

Outline

UPPSALA
UNIVERSITET

Outline

Fill in the following table, no justifications are needed.

Expected time			
	Instruction		
Data structure	Insert	Get-Ith-Element	Contains
Linked List			
Array			
Hash Table[1]			
Binary Search Tree		$\mathcal{O}(n)$	

Make sure you fill in expected time, and *not* worst case. You can assume that the set of elements contains no duplicates, and that any array is allocated with sufficient slack capacity as to hold the additional elements you want to insert. Also assume that the collections are not sorted unless for BST where sortedness is intrinsic to the data structure. State any additional assumptions you make, if anything is unclear.

[1] Assume uniform hashing and collision resolution by chaining.

1) Hermione has written an application to keep track of her many books, where she stores her books in a linked list. However, she complains that it when the running time seems to be quadratic, not linear as she expected, when she prints all book titles. Below the code of Print-All-Book-Titles is given. Explain what the problem is and suggest a solution, either by changing book printing algorithm or changing the data structure used for keeping track of books. Also state the expected running time for book printing when the program is changed according to your suggestion.

```
Print-All-Book-Titles(Books)
1   for i ← 1 to size(Books)
2         do book ← Get-Ith-Element(Books, i)
3            Print book
```

2) The complaint that the printing of all book names is slow is, although correct, not very important. Way more problematic is the fact that checking whether she has a book or not is linear. As this is an operation performed more often than printing all books, she needs to structure her data in such a way that this sort of lookup can be done efficiently. Suggest what data structure she should use if she wants to optimize performance in this regard, and what the expected running time of lookup would be then.

3) After using her application (modified according to your suggestion above) for some time, Hermione has the following two remarks:

- It turns out that Hermione does not do very many lookups either. Her empirical work tells her that she does approximately $\lg n$ lookups every week.
- She is really annoyed by the fact that the list of books that is printed is still not sorted. She of course knows of all sorts of sorting algorithms, but she has a feeling that she should be able to maintain her data in such a way that this is done automagically. She prints her books once every week.

Suggest the data structure that Hermione should use so that the total work per week (lookups and printing, ignore insertions) is minimized, and state the resulting expected asymptotic bound on the weekly computation time.

Outline

Graphs - what are they good for?

Provide a good representation of...

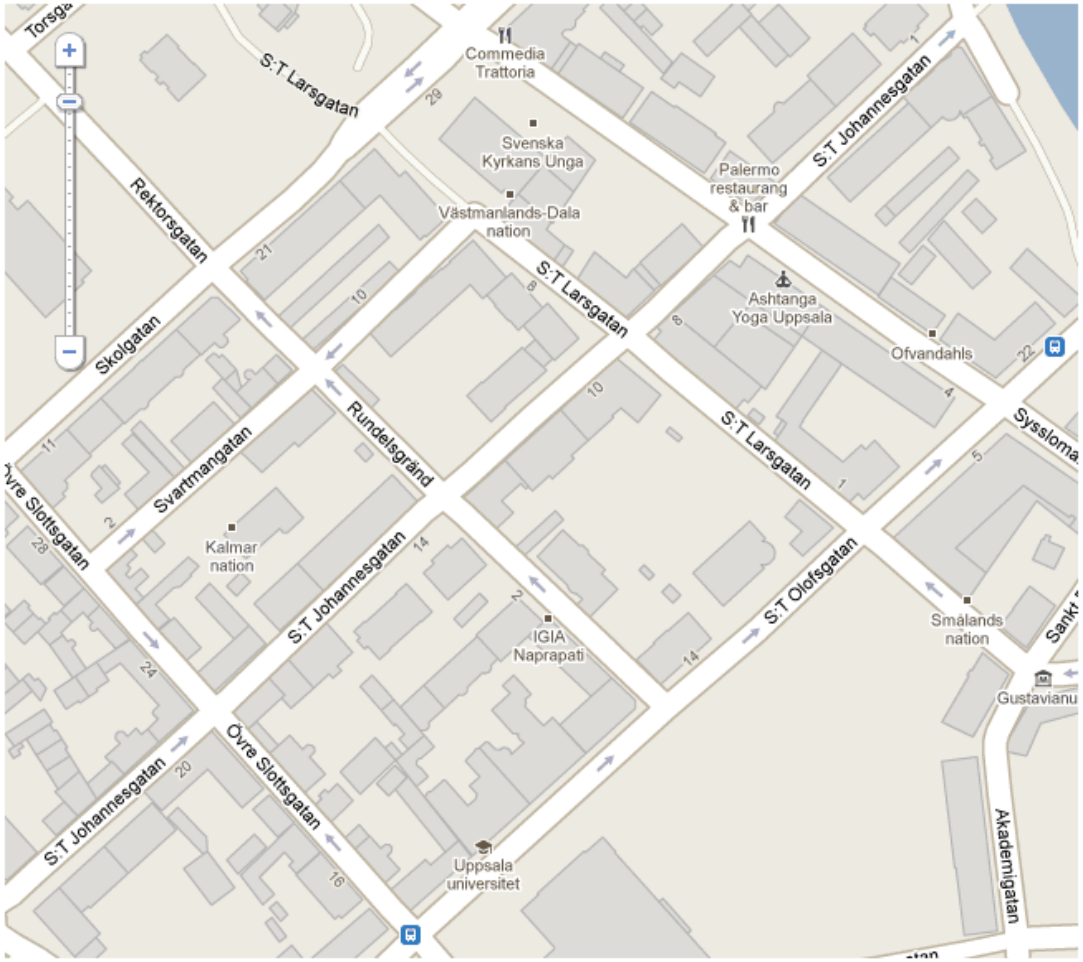
- Data structures (linked lists, tables, trees)
- The internet
- DNA
- A road net
- A sewer system
- The circulation of the blood in the body
- ...

Google uses graphs!

And then?

With the representation, we can isolate interesting properties of the graph, and thus discover interesting properties of the underlying system.

Example: A road system is abstracted as a directed graph in the natural way. What information would computing the strongly connected components of this graph give us?


S:T Larsgatan
Commedia Trattoria
Svenska Kyrkans Unga
Västmanlands-Dala nation
Palermo restaurang & bar
S:T Johannesgatan
Rektorsgatan
Ashtanga Yoga Uppsala
Ofvandahls
Skolgatan
Rundelsgränd
Svartmangatan
Kalmar nation
S:T Olofsgatan
Smålands nation
IGIA Naprapati
Gustavianu
Övre Slottsgatan
Uppsala universitet
Akademigatan


S:T Larsgatan
Rektorsgatan
S:T Johannesgatan
Svenska Kyrkans Unga
Västmanlands-Dala
Ashtanga Yoga Uppsala
Kalmar nation
IGIA Naprapati
Smålands nation
Gustavianu
Akademigatan
Sysslomä
S:t Olofsgat
Uppsala universitet

Whether there is a path from every junction to all other junctions.

Fundamentals: DFS and BFS

- We look a vertex at a time, examining it and discovering its neighbours (moving the frontier).
- The difference lies in how when the neighbours are examined!

DFS

- When examining a node, just tag it as discovered and examine all not yet discovered neighbours first.
- So we examine the nodes using a LIFO policy.

BFS

- When examining a node, put all not yet discovered neighbours on hold and let them "wait for their turn".
- So we examine the nodes using a FIFO policy.

The essence

- DFS uses LIFO, BFS uses FIFO.
- This means they are essentially the same, except that DFS uses a stack and BFS uses a queue.

Note that this glosses over a lot of important points, read the book for more detailed explanation.

Outline

Example exam question

Suppose you have a set of numbers where you over some time period do a constant number of insertions and a linear number of lookups of the maximum. For maintaining your set, of numbers, you have at your disposal a heap and a binary search tree.

a) Assume you are interested in minimizing the total average case runtime over the time period. For each structure, what is the asymptotic upper bound on the average case runtime, and which of the two structures should you choose?

b) Suppose you realize that you are actually interested in the worst case. For each structure, what is the asymptotic upper bound on the worst case runtime, and which of the two structures should you choose now?