Summary of data structures in the course

- Arrays
- Vectors (resizable arrays)
- Linked lists
- Stacks and queues
- Trees (search trees and also heaps)
- Hash tables
- Graphs

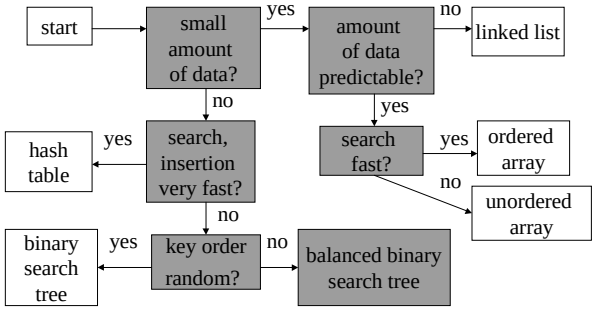
General purpose data structures

- Unordered array
- Ordered array
- Linked list
- Ordered linked list
- Binary search trees
- Balanced binary search trees
- Hash tables

Time complexity of insertion

- Unordered array: O(1)
- Ordered array: O(N)
- Linked list: O(1)
- Ordered linked list: O(N)
- Binary search trees: O(N) worst case, O(log N) on average.
- Balanced binary search trees: O(log N)
- Hash tables: O(1)

Which one to choose (from Lafore's textbook)



Choosing a data structure

- Decision diagrams such as this should be taken with a pinch of salt.
- Given a problem, there are sensible and less sensible choices of a data structure, both from the ease of programming point of view and from efficiency point of view.
- Just like choosing a right tool for the job, some of it is obvious and some of it is down to experience or even to personal preference

Exam revision

- The school's policy is not to provide model answers for exams.
- However answers for formal and informal courseworks are available on-line.

Revision for exams

- Main things tested in the exam
- Exam format
- How to revise

What is tested in the exam

- Knowledge of data structures (e.g. what is a complete binary search tree; give an example; show the result of inserting this value into this tree…)
- Knowledge of algorithms (e.g. give pseudocode or Java code of selection sort)
- Understanding big-Oh notation (e.g. what is the time complexity of this algorithm)

What is tested in the exam

- Given a problem, suggest which algorithms and data structures are appropriate for solving it.

Example: implementing a telephone directory. First need to identify which operations are going to be performed (define ADT) then choose a data structure to store telephones and names so that search etc. is efficient.

"Do we have to write code?"

- Yes
- I will not expect you to implement huge data structures like AVL trees in 30 minutes but something which takes 20-30 lines of Java code.
- If you cannot give proper Java code try to give as detailed pseudocode as possible.

Example: selection sort

- Vague pseudocode: given an array of numbers of length n, loop from i = 0 to i = n-1. Using an inner loop, find the index k of the largest number between arr[0] and arr[i]. Swap this number at position i.
- *Will get you a pass mark.*

Example: selection sort contd.

- Better effort:

Given an array of numbers arr of length n,
loop from i = 0 to i = n-1.

```
    k = 0
    loop from j = 0 to j = i.
       if (arr[j] > arr[k]) k = j
     swap(i,k)
```

Example: selection sort contd.

- Even better, give real Java code when required.
- Please don't write any UserInput routines or any other testing stuff. If you are asked to implement some method, e.g. sorting, just write the code for that method.

Exam Format

- Should attempt **four** out of **six** questions (only the first four will be marked! Cross out the answers you don't want to be marked).
- **Question 1** is a compulsory question (multiple choice, covers the whole course).
- The other three out of five are up to you.

Multiple choice

- Multiple choice this year is straightforward `select one correct option'.
- If you select the right option, you get marks, if not, you get 0 marks for that part (no negative marks!).

How to revise

- Straightforward knowledge questions: lecture notes and any ADS textbook (e.g. Shaffer).
- Choosing appropriate data structures and algorithms: use knowledge of effectiveness and other properties (dynamic vs static) of different data structures.

To sum up

- Do all informal courseworks if you have not done so yet.
- For every algorithm I explained, practice tracing it on some example. (Draw a graph and do Prim's algorithm for it. Draw a B-tree and insert some new elements in it.)
- Do informal and formal courseworks from previous years. They have model answers.