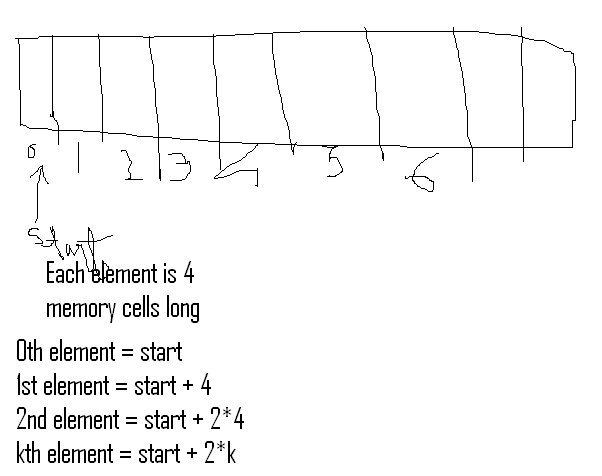
Lecture 9

Binary search is: Ο (log n)

Implementation of a list: done last time

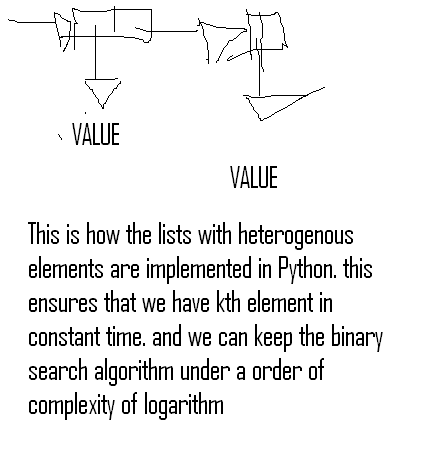
If the list has only integers then:



But if the list has heterogeneous elements: then the normal way to do this is a linked list.

* but in a linked list the implementation time is not constant.
* it is linear access.

in python this is done as:



But for binary search and we need a sorted list.

But is it worth sorting before doing the search?

Can we sort a list in sub-linear time? – no

Can we sort a list in linear time? – probably NOT

How fast can we sort a list? – n log n time. where n is the length of the list.

Linear – n time

Sort and search – n log n + log n

obviously the sort and search is slower.

we can Amortize the cost:

for k searches, the costs are:

linear – k \* n

sort and search – n log n + k n log n

now the sort and search is less.

**SORTING:**

**Selection Sort:**

We will take the smallest element of the list and bring it to the front.

Selection sort relies on LOOP INVARIANT.

LOOP INVARIANT = is a property that is true of this structure everytime through the loop.

loop invariant here is:

* list is split into a prefix and a suffix
* prefix is sorted and suffix is not

and the loop goes on till the prefix is the whole list and the suffix is nothing

order of growth is: Ο (quadratic in the length of the list)

**Bubble sort:**

on the first pass through, the last element is the biggest element on the list.

complexity is: Ο(quadratic in length of the list)

Binary search is the simplest version of DIVIDE AND CONQUER ALGORITHM.