3.1 analysis and big-O

Tuesday, September 13, 2022

Reminder: test 1 in lab next week

TODAY

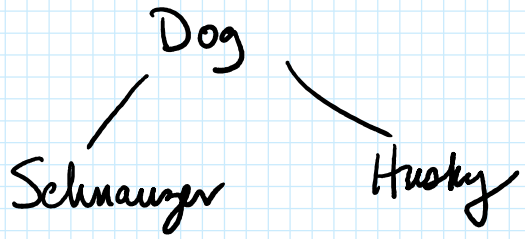
polymorphism and destructors

evaluating algorithms { empirical analysis / theoretical analysis

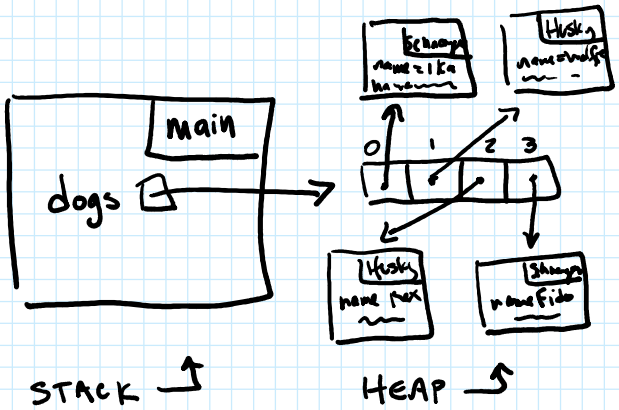
big-O definition, theoretical analysis

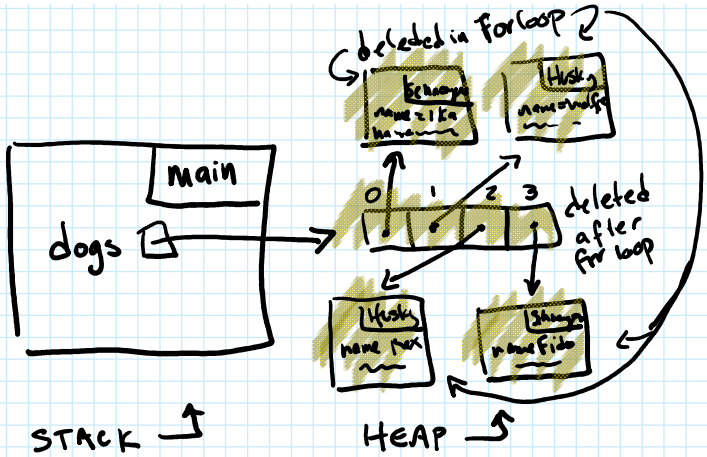
Recap: polymorphism, inheritance, destructors

With an inheritance relationship like



a pointer to the super class type Dog can point to any memory storing an object whose type is a subclass





In lab we'll see

valgrind ./main

again.

problem: bool is_sorted(int* array, int size)

problem: bool is_sorted(int* array, int size)

returns { true, if values are in ascending order
false, if values are NOT in ascending order

ex

```
int a[5]={2, 4, 6, 8, 10};
int b[3]={13, 1, 7};
```

is_sorted(a,5) true!

is_sorted(b,3) false!

Which implementation is "best"?

- fastest time to run
- smallest memory space used
- readable, easy to understand
- flexible, easy to edit
- correct

version 1: correct ✓

version 2: correct ✓

version 3: correct ✓

Empirical Analysis

Want to compare versions to find which is most efficient.

- let's run some experiments and time them
- vary the size of the array

version \ input size	10	100	1000	10,000	50k
1	0,004	0.004	0.009	0.125	2.9
2	0.005	0,003	0.002	0.004	0.004
3	0,006	0.016	0.091	0.857	4.2

overall:

version 2 seems best

versions 1 & 3 are both slower, but 3 is worse than 1 as size increases

notes on empirical analysis

- clock time is not the cleanest measure
  hard to control conditions for equal runs
- can only run on a limited set of values
- must implement each algorithm to test it

THEORETICAL ANALYSIS

- abstract away from a particular language
- analyze how much work the algorithm requires
  - work = # of steps
- count the # of steps in an algorithm
- focus on key steps
  - is_sorted: key step is comparison between elements
- let n represent the size of the problem

Version 1

size = n

i loop goes from i = 0 to i = size - 1
  j loop goes from j = i+1 to j = size - 1
    comparison

How many comparisons are done in total?

$(n-1)+(n-2)+(n-3)+ \ldots + 3 + 2 + 1 = \text{total}$

$1 + 2 + 3 + \ldots + (n-3)+(n-2)+(n-1) = \text{total}$

$n + n + n + \ldots + n + n + n = 2 \cdot \text{total}$

$(n)(n-1) = 2 \cdot \text{total}$ $\quad \text{total} = \frac{n^2}{2} - \frac{n}{2}$

Version 2:

Version 2:

i loop goes from i=0 to size-1

comparison

How many comparisons are done in total?

version 3

i loop goes from i=0 to size-1

10K comparisons

How many comparisons are done in total?