Selecting the kth ordered item from an unordered list


OECD: Ireland's minimum

www.irishtimes.com/business/economy/oecd-ireland-s-minimum-wage-

THE IRISH TIMES

Tue, Feb 9, 2016

NEWS SPORT BUSINESS OPINION LIFE & STYLE CULTURE

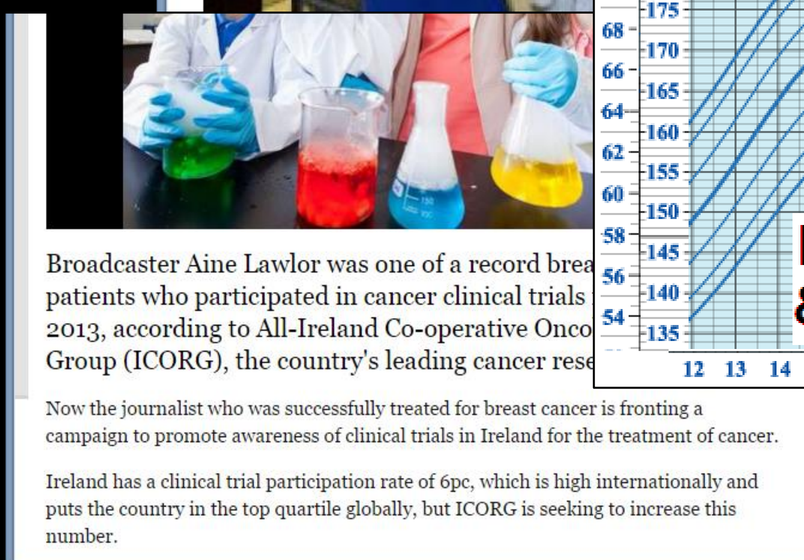
The Economy ❯ Small Business | Budget 2016

OECD: Ireland's minimum wage lags behind median wage

OECD warns time running out for millions of workers at bottom of economic ladder

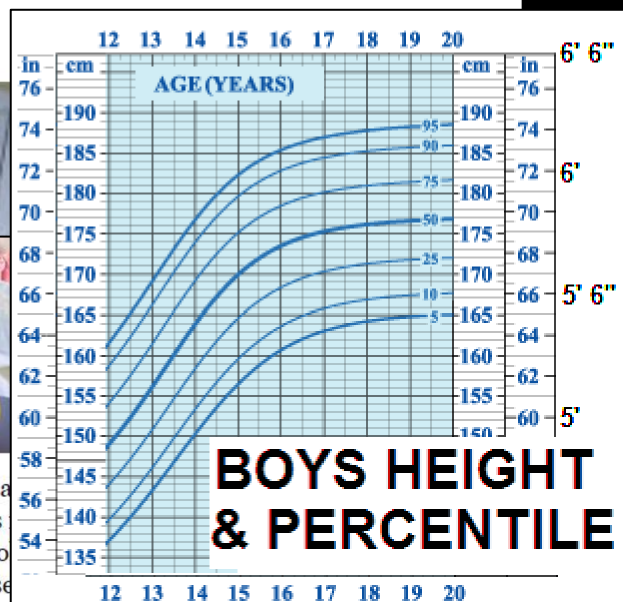
Thu, Jul 9, 2015, 12:20

Erin McGuire


Broadcaster Aine Lawlor was one of a record brea
patients who participated in cancer clinical trials
2013, according to All-Ireland Co-operative Onco
Group (ICORG), the country's leading cancer rese

Now the journalist who was successfully treated for breast cancer is fronting a campaign to promote awareness of clinical trials in Ireland for the treatment of cancer.

Ireland has a clinical trial participation rate of 6pc, which is high internationally and puts the country in the top quartile globally, but ICORG is seeking to increase this number.



A common task in data analytics, scientific computing, social sciences, statistics, economics, etc is identifying *percentiles* in a collection of items.

Given a set of items, the kth percentile is the value below which k% of the items are ranked.

1, 3, 4, 6, 6, 8, 9, 12, 15, 18

70th percentile, because 70% of values come after this point,

10th percentile, because 10% of values come after this point, The *median* is a measure of the average of a collection.

- less influenced by extreme values than the *mean*

The median is the 50th percentile.

The upper *quartile* is the subset of values that are 'better' than the 75th percentile.

The lower *quartile* is the subset of values that are 'poorer' than the 25th percentile.

Given an unsorted list of items, how should we compute the kth percentile? how should we compute the median?

```
6  4  12  9  6  15  3  18  1  8
```

Method 1:

```
repeat k times:
    find the biggest item in the list
    replace that item with None
report the most recent item found
```

Analysis:
Each iteration is O(n), so O(k*n)

For small k << log n, this gives k*O(n) = O(n)
For k ~= log n, this is O(n log n)
For large k >> log n, this is $O(n^2)$

Method 2:

for each element in list:
    add element to a sorted linked list
step through linked list to position k

Analysis:
Worst case is for a sorted list, additions require
$O(1 + 2 + 3 + ... + n) = O(n^2)$
followed by k steps, so $O(n^2)$

Similar analysis for adding to a sorted array-based list, but instead of $O(n^2)$ comparisons, need $O(n^2)$ assignments for a reverse sorted list to shift up each item

Method 3:

sort the list by preferred sorting algorithm
read item in position k

Analysis:
For merge or heap sort, $O(n \log n)$
For quicksort, $O(n^2)$, but $O(n \log n)$ on average

Method 4:

apply quicksort idea to identify element in position k

pick a pivot
sweep through list to gather all items less than, equal to or greater than pivot
if more items 'less than pivot' than k
    recurse on all items less than pivot
else if more items 'less than or equal to pivot' than k
    return pivot
else
    recurse on all items greater than pivot

'decrease and conquer'

Note: don't sort every sublist – all we are doing is picking a pivot from some part of the list, dividing elements from that part, choosing *one* of the divisions, and recursing.

Analysis of method 4:
- worst case – always choose a pivot that decreases the list by just 1 item, so n + (n-1) + (n-2) + .. + (n-(n-(k+1))), which is $O(n^2)$ for small k

- average case

By a similar argument to the one for quicksort, it is possible to show that the average time for a list of length n is O(n).

Practical considerations:

We can implement quickselect in-place, in the same style as in-place quicksort.

Efficient implementation in python is not so obvious ...

Exercise:
- develop versions of quickselect which:
(i) creates lists less, equal and more using python lists and append
(ii) creates lists less, equal and more using 3 list comprehensions
(iii) implements in place
- and algorithms which
(iv) pre-sort with mergesort, heapsort or quicksort, then select
(v) pre-sort with python list sort, then select

Which is fastest on average?

Next lecture

Graphs and Graph Algorithms