CMPT 225 D100 LAB04

TA

TOPICS FOR TODAY

Empirically measuring performance

- Using the chrono library more
- Testing Big-Theta empirically

Performance of Functions

- Given any function, we want them to have **good performance**, which can be measured in two main ways:
  - How much time it needs to produce the output (time performance)
  - How much memory space it needs to produce the output (space performance)
- Here we examine the **time performance**, which can be measured by:
  - Actual time for a function to finish
  - Simple counting of number of steps involved

Empirical Analysis

- Last week:
  - We compared the time taken for:
    - Recurrsion vs Iteration functions
    - Array Object vs static array data structures
- This week:
  - We will compare the time taken for:
    - functions vs their theoretical Big-O?

Big-Theta

- In lecture, we determined the runtime as $T(n)$ by counting simple operations. Assuming that these operations take constant time, then we could say $t(n) \approx cT(n)$, were $t(n)$ is the actual time.
- To determine the Big-Θ, we said that T(n) is $\Theta(g(n))$ if there exists constants **c'** and **c''** such that:
  - **c'**$g(n)$ **< T(n) < c''** $g(n)$
- What does that mean?
  - $t(n) \approx cT(n) \approx cg(n)$ for some constant **c**, such that **c' < c < c''**

Complexity class

- Example:
  - Let $t(n)$ be the time taken for sorting an array of size *n*
  - Let $g(n)$ be our complexity class [ie $\mathbf{\Theta}(g(n))$]
  - Then the relationship is $g(n)$ = c $f(n)$ where c is a constant number

  - Thus, for sufficiently with large n, it should produce $g(n)$ / $f(n)$ = c
    Then we can use this to support our conclusion that $f(n) \in \mathbf{\Theta}(g(n))$

  - This does not <u>prove</u> that $f(n) \in \mathbf{\Theta}(g(n))$ because it does not consider all cases. But this can be useful to support any big-$\mathbf{\Theta}$ guesses

Question: What Big-Θ is $f(n)$?

n	$f(n)$	$\log n / f(n)$	$n/f(n)$	$n \log n \,/ \boldsymbol{f(n)}$	$n^2 / \boldsymbol{f(n)}$
128	0.026	260.9	4772.3	33406.4	610860.8
256	0.034	233.7	7481.45	59851.6	1915251.6
512	0.051	176.2	10026.2	90236.3	5133443.5
1024	0.096	103.8	10630.2	106302.7	10885404
2048	0.192	57.2	10661.3	117274.5	21834397

Question: What Big-$\boldsymbol{\Theta}$ is $f(n)$?

Answer: $\boldsymbol{\Theta}$(n) since that column converges to a near constant value

n	$f(n)$	$\log n / f(n)$	$n / f(n)$	$n \log n \,/\, \boldsymbol{f}(\mathbf{n})$	$n^2 / \boldsymbol{f}(\mathbf{n})$
128	0.026	260.9	4772.3	33406.4	610860.8
256	0.034	233.7	7481.45	59851.6	1915251.6
512	0.051	176.2	10026.2	90236.3	5133443.5
1024	0.096	103.8	10630.2	106302.7	10885404
2048	0.192	57.2	10661.3	117274.5	21834397

Exercise

- The **meausingTimeAgain.cpp** file has a function f that does two things:
  - Fills an array with random values (this is Big-Θ(n))
  - Sorts the array (for now, the is Big-Θ is a mystery)
- Tasks:
  1. Finish the code to print out a table similar to that on the last slide. What is the Big-Θ of function f ? What does that tell us about the Big-Θ of sort? Double check your answer here: https://cplusplus.com/reference/algorithm/sort/
  2. Remove the line: `fillRandom(A,n)` and run the test again. What is the Big-Θ of sort? Why may it be different?

Example Results

n	f(n)	log n	n	n log n	n^2
128	0.044007417	159.0640959	2908.600612	20360.20428	372300.8783
256	0.075251667	106.3099373	3401.917993	27215.34395	870891.0063
512	0.152940416	58.84644645	3347.708954	30129.38058	1714026.984
1024	0.334379084	29.90617679	3062.392503	30623.92503	3135889.923
2048	0.725230458	15.16759242	2823.929935	31063.22928	5783408.507
4096	1.560896459	7.687889822	2624.133059	31489.59671	10748449.01
8192	3.378660334	3.847678877	2424.629643	31520.18536	19862566.04
16384	7.375965208	1.898056675	2221.268612	31097.76057	36393264.94
32768	15.39009521	0.9746528398	2129.161617	31937.42425	69768367.86
65536	33.06055879	0.483960362	1982.301643	31716.82628	0