CSC263 - Week 1, Tutorial 1

Cristyn Howard

Friday, January 12, 2018

Today's topic: Review of running time, proofs of running time.

- T(n); worst-case running time; the maximum number of steps the algorithm T takes on an input of size n
  - t(x); the number of steps taken by the algorithm on specific input x
  - $T(n) = max\{t(x) \quad | \quad |x| = n\}$
- O, Ω, Θ; mathematical concepts that quantify the relations between functions;
  - adopted by computer scientists to abstractly compare running time of different algorithms, without concern for specific implementation details
  - $O(n^2) = \{1, 2, n^2, 2n^2, n^2 + 1, ...\}$, (infinite set)
  - $\Omega(n^2) = \{n^2, 2n^2, 4n^3, ...\}$, (infinite set)
  - $\Theta(n^2) = O(n^2) \cap \Omega(n^2) = \{n^2, 2n^2, ...\}$, (infinite set)
- $T(n) \in O(g(n)) \iff$ for every input of size n, T takes at most $c \times g(n)$ steps, (where $c \in \mathbb{R}$)
- $T(n) \in \Omega(g(n)) \iff$ there is some input of size n for which T takes at least $c \times g(n)$ steps, (where $c \in \mathbb{R}$)

Ex 1:

```
input: A[1...n]
for i = 1 to n:
    for j = 1 to n:
        if A[i] != 1, STOP
```

$O(n^2)$ because the max possible iterations of the inner & outer loop result in n^2 calls of the fourth line of code, which is in constant time

$\Omega(n^2)$ because $\forall\ n, \exists$ array of 1's of size n, which will result in runtime of n^2

Ex 2:

```
Bubblesort(A[1...n]):
last = n, sorted = false;
while(not sorted):
    sorted = true;
    for j=1 to last-1:
        if (A[j] > A[j+1]):
            swap A[j] & A[j+1];
            sorted = false;
    last -= 1;
```

Loop Invariant: at the end of the i^{th} iteration of the while loop, A[last+1 ... n] contains the largest elements in A in sorted order

$T(n) \in O(n^2)$:
- while loop executes at most n times
  (each loop reduces 'last' by 1; when 'last'=1, sorted set true, for loop not executed, while loop stops)
- inner loop executes at most n-1 times
- $n(n-1) \approx n^2$

$T(n) \in \Omega(n^2)$:
- consider reverse sorted array of size n
- for 1^{st} iteration of while loop, for loop does n-1 swaps
- for i^{th} iteration of while loop, for loop does n-i swaps
- total swaps reverse sorted array of size n $= \sum_{i=1}^{n} n - i = \frac{n(n-1)}{2} \in \Theta(n^2)$
- thus for every n, $\exists$ an input array for which the number of steps is at least $\in \Theta(n^2)$