Algorithms and programming
École normale supérieure - Département d'informatique September 28-29, 2017
TD 1: Introduction to algorithms and data structures

---

1 Asymptotic behaviour of functions

SOME USEFUL NOTATION
$O(g(n)) = \{f(n) : \exists c > 0 \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}$
$o(g(n)) = \{f(n) : \forall c > 0\ \exists n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}$
$\Omega(g(n)) = \{f(n) : \exists c > 0, n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\}$
$\Theta(g(n)) = \{f(n) : \exists c_1, c_2 > 0 \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}$

1. For each group of functions, arrange the functions in the group in increasing order of growth. That is, arrange them as a sequence $f_1, f_2, \ldots$ such that $f_1 = O(f_2)$, $f_2 = O(f_3)$, $f_3 = O(f_4), \ldots$. Explain your ordering.
   (a) $f_1 = n$, $f_2 = n^2$, $f_3 = n \log n$, $f_4 = 2^n$, $f_5 = (\log n^2)^2$, $f_6 = 4 \log n$
   (b) $f_1 = \log(\log n)^3$, $f_2 = (\log n)^{3 \log 3n}$, $f_3 = 3^{\log n}$, $f_4 = n^{3^{\log n}}$, $f_5 = (\log \log n)^3$
   (c) $f_1 = 4^{3n}$, $f_2 = 2^{n^4}$, $f_3 = 2^{3^{n+1}}$, $f_4 = 3^{3^n}$, $f_5 = 2^{5n}$
2. Prove that for any real constants a and b, where $b > 0$, holds $(n + a)^b = \Theta(n^b)$.
3. Show that $n! = o(n^n)$ and $\log(n!) = \Theta(n \lg n)$. Is $\lceil \log n \rceil !$ a polynomially bounded function ?
4. Working from the definitions show that for any two functions $f(n)$ and $g(n)$ we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
5. Prove or disprove :
   (a) $f(n) + g(n) = \Theta(\min(f(n), g(n)))$
   (b) $f(n) = O((f(n))^2)$
   (c) $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$
   (d) $f(n) = \Theta(f(n/2))$
   (e) $f(n) + o(f(n)) = \Theta(f(n))$

2 Elementary data structures

1. Explain how to implement two stacks using one array. Operations POP and PUSH must take $O(1)$ time.
2. Implement a queue using a linked list. Analyse the performance of your implementation.
3. Implement a stack using a linked list.
4. Develop two implementations of a stack using two queues. Analyse their performance.

5. Develop a (non-recursive) procedure with running time $\Theta(n)$ that reverses the order of elements in a linked list. The procedure must use $O(1)$ extra space.
6. Let T be a rooted binary tree, where each of its nodes is assigned an integer key. Assume that each node stores pointers to the parent node, to the left child and to the right child. Develop a (non-recursive) linear-time procedure that prints out all the keys. Use a stack.
7. Professor Bunyan thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets : A, the keys to the left of the search path ; B, the keys on the search path ; and C, the keys to the right of the search path. Professor Bunyan claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a smallest possible counterexample to the professor's claim.
8. Let T be a binary search tree, let x be a leaf node, and let y be its parent. Show that $key[y]$ is either the smallest key in T larger than $key[x]$ or the largest key in the tree smaller than $key[x]$.

3 Amortised analysis

1. A sequence of stack operations is performed on a stack whose size never exceeds k, After every k operations, a copy of the entire stack is made for backup purposes. Show that the cost of n stack operations, including copying the stack, is $O(n)$ by assigning suitable amortized costs to the various stack operations.
2. Let A be a binary array of length n. Develop an algorithm that finds the longest segment of S such that the sum of elements in it equals exactly k.