

Solutions to Practice Problems 1

Solution 1:

$$\begin{array}{lll}
\text{(a)} & O((2^{1/3})^n) & \prec O((3/2)^n) \prec O((3^{1/2})^n) \\
\text{(b)} & O(\log n) & = O(\log n) = O(\log n) \\
\text{(c)} & O(n) & \prec O(n^2) = O(n^2) \\
\text{(d)} & O(n^2) & \prec O(n^3) = O(n^3) \\
\text{(e)} & O(1) & \prec O(n) = O(n)
\end{array}$$

- (a) We use the fact that $a^{bc} = (a^b)^c$, and hence $(3/2)^n = 1.5^n$, $3^{(n/2)} = (3^{1/2})^n \approx 1.73^n$, and $2^{(n/3)} = (2^{1/3})^n \approx 1.26^n$. The big-O expressions are $O((3/2)^n)$, $O((3^{1/2})^n)$, and $O((2^{1/3})^n)$. All of these are exponential, if $x < y$, then $y^n/x^n = (y/x)^n$, therefore $x^n \prec y^n$. The order is $O((2^{1/3})^n) \prec O((3/2)^n) \prec O((3^{1/2})^n)$.
- (b) Use the fact that $\log_a b = \log_c b / (\log_c a)$, for any constant base $x > 1$, $\log_x n = \log n / \log x$ is $O(\log n)$, since $1/\log x$ is a constant. Therefore the first two are $O(\log n)$. For the third one, $\log(n^2) = 2 \log n = O(\log n)$. All three have the same big O.
- (c) Use $\lg = \log_2$. $2^{\lg n} = n = O(n)$, $n^{\lg 4} = n^2 = O(n^2)$, and finally $2^{2 \lg n} = (2^{\lg n})^2 = n^2 = O(n^2)$.
- (d) For sufficiently large n , $50n^2 < n^3$. $\max(50n^2, n^3) = n^3 = O(n^3)$, $\min(50n^2, n^3) = 50n^2 = O(n^2)$. For the second one, $50n^2 + n^3 = O(n^3)$.
- (e) $n + 100 = O(n)$, $10^{100} = O(1)$, $100n = O(n)$.

Solution 2:

- (a) We can use the summation formula of geometric series when $0 < c < 1$: $\sum_{i=0}^n c^i = \frac{1-c^{n+1}}{1-c} < \frac{1}{1-c}$. We found a constant upper bound of the function, so it is $O(1)$.
- (b) When $c = 1$, every term is 1, so $f(n) = n + 1 = O(n)$.
- (c) When $c > 1$, using the summation formula for geometric series, $\sum_{i=0}^n c^i = \frac{c^{n+1}-1}{c-1} = O(c^n)$. Intuitively, the largest term c^n dominates the growth rate.

Solution 3: The outer loop cuts i in half each time, and for a given i the inner loop does i iterations. So i takes the values $n, n/2, n/4, \dots, 1$ (about $\lg n$ values), and the total amount of work is

$$n + \frac{n}{2} + \frac{n}{4} + \dots + 1 = \left(1 + \frac{1}{2} + \frac{1}{4} + \dots\right) n \leq 2n,$$

using the fact that the geometric series $\sum_{k \geq 0} (1/2)^k$ sums to 2 (the finite sum here is strictly less than 2). Therefore the overall time complexity is $O(n)$.

Solution 4:

- (a) The edges of the graph are $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow F$, $B \rightarrow C$, $D \rightarrow C$, $F \rightarrow E$, $F \rightarrow G$, and $G \rightarrow A$. Adjacency matrix:

	A	B	C	D	E	F	G
A	0	1	1	0	0	1	0
B	0	0	1	0	0	0	0
C	0	0	0	0	0	0	0
D	0	0	1	0	0	0	0
E	0	0	0	0	0	0	0
F	0	0	0	0	1	0	1
G	1	0	0	0	0	0	0

Adjacency list:

A : $B \rightarrow C \rightarrow F$
 B : C
 C : (empty)
 D : C
 E : (empty)
 F : $E \rightarrow G$
 G : A

- (b) Visit order:

A, B, C, F, E, G, D .

Note that D is not reachable from A , so it is never visited by this search. If we run a full DFS over the whole graph, we would restart at the remaining node D , giving the overall order A, B, C, F, E, G, D .

- (c) The resulting DFS forest is shown below.

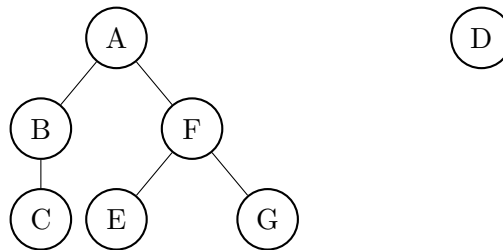


Figure 1: DFS forest. The second tree consists of the single node D .

$A \rightarrow B$ tree edge
 $B \rightarrow C$ tree edge
 $A \rightarrow F$ tree edge
 $F \rightarrow E$ tree edge
 $F \rightarrow G$ tree edge
 $A \rightarrow C$ forward edge
 $G \rightarrow A$ back edge
 $D \rightarrow C$ cross edge

Solution 5: The maximum number of edges is

$$n_1n_2 + n_1n_3 + n_2n_3.$$

Every edge must have its two endpoints in two different parts, so each edge falls into exactly one of three categories: between V_1 and V_2 , between V_1 and V_3 , or between V_2 and V_3 . There are at most n_1n_2 pairs of the first kind, at most n_1n_3 of the second, and at most n_2n_3 of the third, which gives the upper bound.

Solution 6: The statement is **true**. We prove it by contradiction.

Suppose G has n nodes, every node has degree at least $n/2$, but G is not connected. Then G has at least two connected components. Let C be a smallest component and let $k = |C|$. Since the components are disjoint and there are at least two of them, the smallest one can contain at most half of the nodes, so $k \leq n/2$.

Take any node $v \in C$. All edges incident to v stay inside C (an edge leaving C would merge C with another component), and v can be adjacent only to the other $k - 1$ nodes of C . Hence

$$\deg(v) \leq k - 1 \leq \frac{n}{2} - 1 < \frac{n}{2},$$

which contradicts the assumption that every node has degree at least $n/2$. Therefore G must be connected.