Algorithmic Thinking
Luay Nakhleh

Pseudo-code: Syntax

Pseudo-code is a high-level, abstract description of an algorithm that is intended to be read by humans and subsequently turned into a program in a programming language of choice. As such, pseudo-code provides the details needed to understand the algorithm, but omits many details that are left to the programmer to implement, since those may vary depending on the programming language and the choice of implementation.

1 General

- $v \leftarrow a$: assign value a to variable v
- $v = w$: testing whether v equals w (the value of which is True if v equals w, and False otherwise)
- $v \neq w$: testing whether v is not equal to w (the value of which is True if v is not equal to w, an False otherwise)
- a or b: a or b (which is true if and only if at least one of a and b is true)
- a and b: a and b (which is true if and only if both of a and b are true)
- $random(a, b)$: returns a random number in $[a, b)$ (that is, a may be returned, but not b)
- **Break**: breaks out of a loop
- **if** c **then**
  ...
  else
  ...

2 Loops

- **for** $i \leftarrow a$ **to** b **do**: loop over the values of i that start at a (integer) and end at b (integer), while incrementing the value by 1 after each iteration
- **while** c: loop while condition c is true

3 Functions

- **return**: returning a value from a function
- $out \leftarrow \mathbf{foo}(params)$: call function **foo** on its parameters $params$ and store its output in out. For example: $distances \leftarrow \mathbf{BFS}(g, v)$.

4 Arrays, lists, and matrices

Let A be an array, L be a list, and M be a matrix.

- $A[0..n-1]$: Array A has n elements indexed at $0, 1, \ldots, n-1$
- $L[0..n-1]$: List L has n elements indexed at $0, 1, \ldots, n-1$
- $M[0..n-1, 0..m-1]$: Matrix M has n rows, indexed at $0, 1, \ldots, n-1$, and m columns indexed at $0, 1, \ldots, m-1$
- $A[i]$: the element of array A at position i
- $L[i]$: the element of list L at position i
- $M[i, j]$: the element of matrix M at position $[i, j]$
- $Copy(L[i..j], L'[p, q])$: assign $L'[p]$ to $L[i]$, $L'[p+1]$ to $L[i+1]$, ... , $L'[q]$ t $L[j]$ (it is assumed here that $j - i = q - p$)

5 Set notation

Let A, B, and C be three sets.

- $\mathbb{R}$: the set of all real numbers
- $\mathbb{Z}$: the set of all integers $(\ldots, -2, -1, 0, 1, 2, \ldots)$
- $\mathbb{N}$: the set of all natural numbers $(1, 2, \ldots, ...)$
- $\mathbb{R}^+$: the set of all positive real numbers (0 is not included)
- $\mathbb{Z}^+$: the set of all positive integers (0 is not included)
- $|A|$: the size (number of elements) of set A
- $a :\in A$: let a be an arbitrary element of A
- $a \in A$: a is an element of A
- $a \notin A$: a is not an element of A
- $A \cup B$: union of A and B
- $A \cap B$: intersection of A and B
- $A \setminus B$: the set difference between A and B (also, $A - B$)
- **foreach** $A \subseteq B$: looping over each subset A of B (including B itself)
- **foreach** $A \subset B$: looping over each subset A of B (excluding B itself)
- **foreach** non-empty $A \subset B$: looping over each subset A of B (excluding $\emptyset$ and B)
- **foreach** $A \subseteq B$ of size k: looping over each subset A of B such that $|A| = k$
- $Perm(A)$: the set of all permutations of the elements of A
- $Random(A, m, p)$: return a random set of m elements from A, where the probability of choosing element $x \in A$ is given by p (for example, $Random(A, m, 1/|A|)$ returns a random set of m elements from A assuming a uniform distribution. Another example: $Random(V, m, deg(v)/(2|E|))$ returns a random set of m nodes from set V of nodes, where the probability of choosing a node is proportional to the node's degree, or, more precisely, it equals the degree of that node normalized by the sum of the degrees of all nodes)

- $\text{argmin}_{x \in A} w_x$: the element of set A whose weight is minimum among all elements of A (it is assumed that each element x in set A has weight w_x, and more than one element has the minimum weight, then one of them is returned arbitrarily)
- $\text{argmax}_{x \in A} w_x$: similar to argmin, but returns the element of A with maximum weight

6 Data structures

- $queue()$: returns an empty queue
- $enqueue(Q, a)$: puts element a in the queue Q
- $dequeue(Q)$: returns the element at the head of the queue Q

7 Graph-theoretic notation

- $V(g)$: the set of nodes of graph g
- $E(g)$: the set of edges of graph g
- $n_g(v)$: the set of neighbors of node v in undirected graph g. If g is clear from the context, we may drop the subscript (that is, just use $n(v)$)
- $deg_g(v)$: the quantity $|n_g(v)|$. If g is clear from the context, we may drop the subscript
- $in_g(v)$: the set of nodes from which there are edges to node v in directed graph g. If g is clear from the context, we may drop the subscript (that is, just use $in(v)$)
- $out_g(v)$: the set of nodes to which there are edges from node v in directed graph g. If g is clear from the context, we may drop the subscript (that is, just use $out(v)$)
- $indeg_g(v)$: the quantity $|in_g(v)|$. If g is clear from the context, we may drop the subscript
- $outdeg_g(v)$: the quantity $|out_g(v)|$. If g is clear from the context, we may drop the subscript
- $w_g(e)$: the weight of edge e in graph g. If g is clear from the context, we may drop the subscript (that is, just use $w(e)$)
- $\text{argmin}_e(A, B)$: the edge whose one endpoint is in set A of nodes and the other endpoint is in set B, and has minimum weight of all such edges (it is assumed that $A, B \subseteq V(g)$ for some graph g, and $e \in E(g)$)
- $nodes_g(E')$: the set of nodes that are endpoints of at least one of the edges in E' in graph g. If g is clear from the context, we may drop the subscript
- $edges_g(V')$: the set of edges each of which has at least one of its endpoints in set V' in graph g. If g is clear from the context, we may drop the subscript
- **foreach** $v \in V(g)$: looping over all nodes in graph g.
- **foreach** $e \in E(g)$: looping over all edges in graph g.

8 String and sequence notation

Let w be a string. We assume that the letters of w are indexed starting at 0

- $|w|$: the length of string/sequence w (that is, the number of letters in w)
- w_i: letter at position i in string w. It is assumed that $0 \leq i \leq |w|$