Example of how to use Algorithm2e

Robert Woodward

September 5, 2019

Below we illustrate the formatting as pseudo code of some sample of simple algorithms. The goal is not to entice you to use LaTeX for formatting your algorithms as currently the best possible formatting tool for algorithms. Please carefully check the source files and learn how to use this style. Importantly:

- Always state your input
- State the output if any
- Always number your lines for quick referral.
- Always declare and initialize your local variables
- Always use \gets for assignments
- Always end with "return" even when not returning any values
- Use common functions and operands such as UNION, POWERSET, etc. as often as needed, unless you are asked to define them.

Algorithm 2 will find the maximum element in a finite sequence (Slide 14 in Class Slides).

Algorithm 3 is a greedy change-making algorithm (Slide 19 in Class Slides).

Algorithm 4 and Algorithm 5 will find the first duplicate element in a sequence of integers.

Algorithm 1: Hungarian Algorithm

Input: A complete bipartite graph $G = (S, T; E)$ with nonnegative cost $c : E \to \mathbb{R}_{\geq 0}$

Output: The perfect matching M with a minimum total cost

`// Initialization`

$y(s) \leftarrow 0 \quad \forall s \in Z \cap S$
All edges are oriented from S to T
while $|M| < |S|$ **do**
    if $R_T \cap Z \neq \emptyset$ **then**
        `// operation1`
        reverse the orientation of a directed path consisting of only tight edges from R_S to R_T
    else
        `// operation2`
        $\triangle \equiv \min\{c(i,j) - y(i) - y(j) \mid i \in Z \cap S, j \in T \setminus Z\}$
        $y(s) \leftarrow y(s) + \triangle \quad \forall s \in Z \cap S$
        $y(t) \leftarrow y(t) - \triangle \quad \forall t \in Z \cap T$
return M

Algorithm 2: MAX finds the maximum number

Input: A finite set $A = \{a_1, a_2, \ldots, a_n\}$ of integers

Output: The largest element in the set

$max \leftarrow a_1$
for $i \leftarrow 2$ ***to*** n **do**
    if $a_i > max$ **then**
        $max \leftarrow a_i$
return max

Algorithm 3: CHANGE Makes change using the smallest number of coins

Input: A set $C = \{c_1, c_2, \ldots, c_r\}$ of denominations of coins, where $c_i > c_2 > \ldots > c_r$ and a positive number n

Output: A list of coins $d_1, d_2, \ldots, d_k$, such that $\sum_{i=1}^{k} d_i = n$ and k is minimized

$C \leftarrow \emptyset$
for $i \leftarrow 1$ ***to*** r **do**
    while $n \geq c_i$ **do**
        $C \leftarrow C \cup \{c_i\}$
        $n \leftarrow n - c_i$
return C

Algorithm 4: FindDuplicate

Input: A sequence of integers $\langle a_1, a_2, \ldots, a_n \rangle$
Output: The index of first location witht he same value as in a previous location in the sequence

```
location ← 0
i ← 2
while i ≤ n and location = 0 do
    j ← 1
    while j < i and location = 0 do
        if a_i = a_j then
            location ← i
        else
            j ← j + 1
    i ← i + 1
return location
```

Algorithm 5: FindDuplicate2

Input: A sequence of integers $\langle a_1, a_2, \ldots, a_n \rangle$
Output: The index of first location witht he same value as in a previous location in the sequence

```
location ← 0
i ← 2
while i ≤ n ∧ location = 0 do
    j ← 1
    while j < i ∧ location = 0 do
        if a_i = a_j then  location ← i

        else  j ← j + 1

    i ← i + 1
return location
```