伪代码简要教程

- 伪代码基本概念
- 伪代码基本规范
- 伪代码实例

伪代码的概念

- 伪代码（pseudocode），又称为虚拟代码，是高层次描述算法的一种方法。它不是一种现实存在的编程语言；它可能综合使用多种编程语言的语法、保留字，甚至会用到自然语言。
- 人们在用不同的编程语言实现同一个算法时意识到，他们的实现（注意：这里是实现，不是功能）很不同。尤其是理解那些不熟悉的编程语言实现的功能时可能很困难，这样伪代码就应运而生了。

伪代码必须结构清晰、代码简单、可读性好，并且类似自然语言。

伪代码的概念

伪代码只是像流程图一样用在程序设计的初期，帮助写出程序流程。简单的程序一般都不用写流程、写思路，但是复杂的代码，最好还是把流程写下来，总体上去考虑整个功能如何实现。写完以后不仅可以用来作为以后测试，维护的基础，还可用来与他人交流，但是，如果把全部的东西写下来必定会浪费很多时间，那么这个时候可以采用伪代码方式，比如：

IF 9点以前 Then

    起床洗漱、早餐、送孩子上学

Else If 9点到18点 Then

    公司上班

Else 在家辅导孩子作业

伪代码基本规范

1. 在伪代码中，每一条指令占一行。

2. 书写上的缩进表示程序中的分支结构，同一模块中的语句具有相同的缩进量，次一级模块的语句相对与其父级模块的语句缩进。

3. 在伪代码中，变量不需要声明，定义变量的语句不用写出来，但要在注释中给出。

基本规范——算法名称

- 过程（Procedure）：执行一系列操作，不需要返回操作的结果，无返回数据。
- 函数（Function）：执行一系列的操作后，要将操作的结果返回，有返回数据。

书写规则

过程 <算法名>([<参数列表>])

函数 <算法名>([<参数列表>])

示例

函数 Greedy-Activity-Selector(S, F)

    指令序列

    Return A

基本规范——输入/输出

书写规则

Input：[<输入参数>]

Output：[<输出参数>]

示例

Input：活动的开始，结束时间数组S, F，假设$f_1 \le f_2 \le \ldots \le f_n$已排序

Output：选择的活动集合A

函数 Greedy-Activity-Selector(S, F)

　　指令序列

　　Return A

基本规范——指令表示

指令：在算法中的某些句子或子任务可以用文字来叙述。这样做的目的就是为了避免因那些与主要问题无关的细节使算法本身杂乱无章。

示例

insert all items in table[T] into new-table;

u ← ExtractMin(Q)

伪代码——表达式

- 算数表达式可以用通常的算数运算符(+, -, *, / 以及表示幂的^)。
- 逻辑表达式可以使用关系运算符=、≠、≤、≥、<、> 以及逻辑运算符与(and)，或(or)，非(not)。

示例

- While i<length[A] and A[i]=1 Do
- If i=0 or j=0
- S ← S ∪ {u}

基本规范——赋值语句

书写规则： a ← b

这里a是变量，b是算数表达式、逻辑表达式或指针表达式，语句的含义是将b的值赋给a。

示例

```
i ← 0
k ← k - 1
D[i,j] ← D[i,k] + D[k,j];
```

基本规范——分支选择

书写规则

➢**单分支**

If <条件> Then
　　指令序列

➢**多分支**

If <条件> Then
　　指令序列
Else
　　指令序列

示例

```
If x[i] = y[j] Then
        C[i,j] ← C[i-1,j-1] + 1;
        B[i,j] ← “↖”;
Else If C[i-1,j] ≥ C[i,j-1] Then
        C[i,j] ← C[i-1,j] + 1;
        B[i,j] ← “↑”;
Else
        C[i,j] ← C[i,j-1] + 1;
        B[i,j] ← “←”;
```

基本规范——循环语句

书写规则

- **计数式循环**

For 变量 ← 初值 To 终值 Do

    指令序列

- **条件式循环**

While <条件> Do

    指令序列

示例

For i ← 1 To m Do

    C[i,0] ← 0;

For each v ∈ G.V Do

    v.d ← ∞;

While i<length[A] and A[i]=1 Do

    A[i] ← 0;

    i ← i+1;

基本规范——数组元素

书写规则：数组元素的存取用数组名后跟“[下标]”表示。符号“…”用来指示数组中值的范围。

示例

A[j]指示数组A的第j个元素。
A[1…n]表示含元素A[1], A[2], … , A[n]的数组。

A[i,j]表示二维数组A的第i行第j列的元素。

伪代码实例

矩阵链乘法

Input: 矩阵连p，p中存的各个矩阵的行数列数

Output: 完成矩阵链乘的最小乘法数矩阵m

过程：Matrix-Chain-Order(p)

```
1.  n ← length(p) - 1;
2.  For i ← 1 To n Do
3.      m[i, i] ← 0;
4.  For l ← 2 To n Do    /* 计算第l对角线 */
5.      For i ← 1 To n - l + 1 Do
6.          j ← i + l - 1;
7.          m[i, j] ← ∞;
8.          For k ← i To j-1 Do    /* 计算m[i,j] */
9.              q ← m[i, k] + m[k+1, j] + p[i-1]*p[k]*p[j]
10.             If q < m[i, j] Then
11.                 m[i,j] ←  q;
12. Return m
```

伪代码实例

Algorithm 1 Find factual discrepancies in a cluster.

Input: Cluster of documents $\mathcal{D}$ with spans $\mathcal{S}$, and NLI model.
Output: Sorted spans by discrepancy likelihood.

$\omega_{i,j} \leftarrow 0 \quad \forall$ span j in document i
for $D_i \in \mathcal{D}$ **do**
  for $S_j \in D_i$ **do**
    $X_{\text{hyp}} \leftarrow S_{i,j}; \quad \Omega \leftarrow \{\}$
    for $D_k \in \mathcal{D} \setminus D_i$ **do**
      $\Gamma \leftarrow \{\}$
      for $S_{k,l} \in D_k$ **do**
        $p_c \leftarrow \text{SENTLI}(X_{\text{hyp}}, S_{k,l})[c]$
        $\Gamma \leftarrow \Gamma \bigcup \{p_c\}$
      $\Omega \leftarrow \Omega \cup \max(\Gamma)$
    $\omega_{i,j} \leftarrow \text{mean}(\Omega)$
return $\mathcal{S}$ sorted by respective ω

Schuster T, Chen S, Buthpitiya S, et al. Stretching Sentence-pair NLI Models to Reason over Long Documents and Clusters[J]. arXiv preprint arXiv:2204.07447, 2022.

伪代码实例

Algorithm 1 Self-Adversarial Learning With Comparative Discriminator

Require: Generator G_θ; comparative discriminator D_ϕ; samples of real sentences S_+; self-adversarial learning step g; discriminator step k; memory buffer $\mathcal{M}$ for the previous generated samples

Pretrain G_θ using MLE on S_+
Generate samples with G_θ and store them into $\mathcal{M}$
repeat
  for k steps **do**
    Collect a mini-batch of balanced sample pairs $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ from $\mathcal{M} \cup S_+$
    Update D_ϕ via Eq (2)
  end for
  for g steps **do**
    Generate a mini-batch of samples $\boldsymbol{x}_g \sim G_\theta$
    Collect a mini-batch of reference samples $\boldsymbol{x}_r$ from $\mathcal{M}$
    Update G_θ via Eq (6)
  end for
  Update $\mathcal{M}$ with G_θ
until Convergence

Zhou W, Ge T, Xu K, et al. Self-Adversarial Learning with Comparative Discrimination for Text Generation[C]//International Conference on Learning Representations. 2019.