Algorithm 1: How to write algorithms

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
    read current;
    if *understand* **then**
        go to next section;
        current section becomes this one;
    else
        go back to the beginning of current section;
    end
end

Algorithm 2: Simulation-optimization heuristic

Data: current period t, initial inventory I_{t-1}, initial capital B_{t-1}, demand samples
Result: Optimal order quantity Q_t^*
$r \leftarrow t$;
$\Delta B^* \leftarrow -\infty$;
while $\Delta B \leq \Delta B^*$ *and* $r \leq T$ **do**
    $Q \leftarrow \arg\max_{Q \geq 0} \Delta B_{t,r}^{Q}(I_{t-1}, B_{t-1})$;
    $\Delta B \leftarrow \Delta B_{t,r}^{Q}(I_{t-1}, B_{t-1})/(r - t + 1)$;
    if $\Delta B \geq \Delta B^*$ **then**
        $Q^* \leftarrow Q$;
        $\Delta B^* \leftarrow \Delta B$;
    end
    $r \leftarrow r + 1$;
end

Algorithm 3: How to write algorithms

Data: this text
Result: how to write algorithm with LaTeX2e

```
1  initialization;
2  while not at end of this document do
3  |  read current;
4  |  repeat
5  |  |  do these things;
6  |  until this end condition;
7  |  if understand then
8  |  |  go to next section;
9  |  |  current section becomes this one;
10 |  else
11 |  |  go back to the beginning of current section;
12 |  end
13 |  do
14 |  |  do these things;
15 |  while this end condition;
16 end
```

Algorithm 4: disjoint decomposition

input : A bitmap Im of size $w \times l$
output: A partition of the bitmap

```
1  special treatment of the first line;
2  for i ← 2 to l do
3  |  special treatment of the first element of line i;
4  |  for j ← 2 to w do
5  |  |  left ←FindCompress(Im[i, j − 1]);
6  |  |  up ← FindCompress(Im[i − 1, ]);
7  |  |  this ← FindCompress(Im[i, j]);
8  |  |  if left compatible with this then // O(left,this)==1
9  |  |  |  if left < this then Union(left,this);
10 |  |  |  else Union(this,left);
11 |  |  end
12 |  |  if up compatible with this then                  // O(up,this)==1
13 |  |  |  if up < this then Union(up,this);
   |  |  |  // this is put under up to keep tree as flat as
   |  |  |     possible
14 |  |  |  else Union(this,up);
   |  |  |  // this linked to up
15 |  |  end
16 |  end
17 |  foreach element e of the line i do FindCompress(p);
18 end
```

Algorithm 5: Calculate $y = x^n$

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  if $n < 0$ **then**
    $X \leftarrow 1/x$
    $N \leftarrow -n$
  else
    $X \leftarrow x$
    $N \leftarrow n$
  end if
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow N/2$
    else $\{N$ is odd$\}$
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while