EXERCISE 18.1. (1) The breadth-first search algorithm has a predictable running time, because the representative operation is the number of arc scannings, and each arc is scanned exactly once during the course of the search. The algorithm thus runs in $\theta(m)$ time.

(2) The original implementation of Dijkstra's algorithm (given in Figure 4.6) has a predictable running time because the representative operation is the number of node scannings in the node selection operation which is exactly $n(n-1)/2$. Hence the algorithm has a predictable running time.

(3) In Dial's implementation of Dijkstra's algorithm, the set of representative operations is the number of arc scannings (which is $m$) and the number of bucket scannings, which differs from instance to instance. The algorithm does not have a predictable running time, because instances of the algorithm may exist in which as large as $nC$ or as small as $n$ buckets are scanned during the course of the algorithm.

(4) The radix heap implementation of Dijkstra's algorithm has three representative operations: (i) the arc scans in the distance updates which are $m$; (ii) the number of buckets scanned to place nodes during the distance updates and redistribution of ranges which are $\theta(n \log(nC))$ (because each node starts at the last bucket and finally ends up in the first bucket); and (iii) the number of buckets whose ranges change during redistributions (and there are at most $n \log(nC)$ such operations, but might vary from instance to instance). The running time of the algorithm will be within a constant factor of the worst-case running time of $O(m+n \log(nC))$ and, hence, the algorithm has a predictable running time.

(5) The $O(nm)$ time negative-cycle detection algorithm does not have a predictable running time because the algorithm terminates as soon as a negative cycle is detected. It is possible that a negative cycle may be detected much before $O(nm)$ arc scannings have been performed.

(6) The $O(nm)$ time minimum mean cycle algorithm has a predictable running time. This is because when we calculate $d_{i+1}(-)$ values from the $d_i(-)$ values, each arc in the network has to be scanned exactly once. Thus, calculations of $d_i(-)$ values such that $1 \leq i \leq n$ requires exactly $\theta(nm)$ steps.

EXERCISE 18.3. (1) The representative operations are: (a) the number of flow changes on an arc due to an augmentation; (b) the number of arc scannings. The number of relabel operations, retreat steps and bad advance steps (which are cancelled later by retreat operations) can be charged to operations in (b), while the number of good advance steps can be charged to (a).

(2) The representative operations are: (a) the number of pushes; and (b) the number of arc scannings during the relabel steps. Notice that the number of arc scanning to identify admissible arcs can be charged to the operations in (b).

(3) The representative operations are: (a) the number of node scannings due to the preprocessing in each scaling phase, which equals $n \log U$; (b) the number of arc scanning during the relabel steps; and (c) the number of pushes.
EXERCISE 18.5. (1) The representative operations for Kruskal’s algorithm are: (a) the number of arcs scanned in the decreasing order of the arc lengths; and (b) the number of nodes examined to merge the lists. Here we assume that arcs are already sorted.

(2) The representative operations for Sollin’s algorithm are: (a) the number of arcs scanned to find the nearest neighbors (which are m times the number of iterations); and (b) the number of arcs scanned while merging the trees.

EXERCISE 18.7. Solutions of Exercises 18.1 and 18.2 give the representative operations for the algorithms. All other nonrepresentative operations performed by the algorithms are dominated by at least one representative operation.

EXERCISE 18.9. Solution omitted.