Chapter-07
Run-Time Environments

Slide Courtesy: Moumita Sarker | Modified by: Zinia Sultana

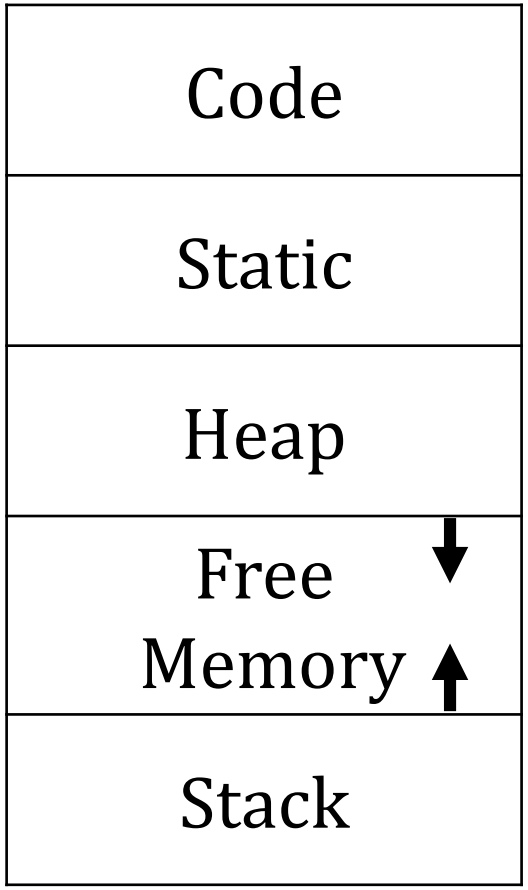
Outline

- Storage Organization
  - Static Versus Dynamic Storage Allocation
- Stack Allocation of Space
  - Activation Trees
  - Activation Records
  - Calling Sequences
  - Variable-Length Data on the Stack

Storage Organization

- From the perspective of the compiler writer, the executing target program runs in its own logical address space in which each program value has a location
- The management and organization of this logical address space is shared between the compiler, operating system and target machine
- The operating system maps the logical addresses into physical addresses which are usually spread throughout memory

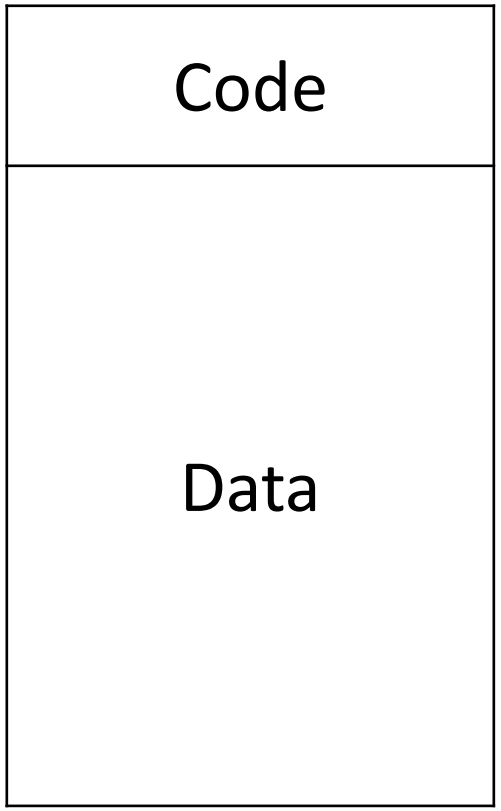
Storage Organization (Cont…)



- The run-time representation of an object program in the logical address space consists of data and program areas as shown in figure
- A compiler for a language like **C++** on an operating system like **Linux** might subdivide memory in this way

Typical Subdivision of Runtime Memory into Code and Data Areas

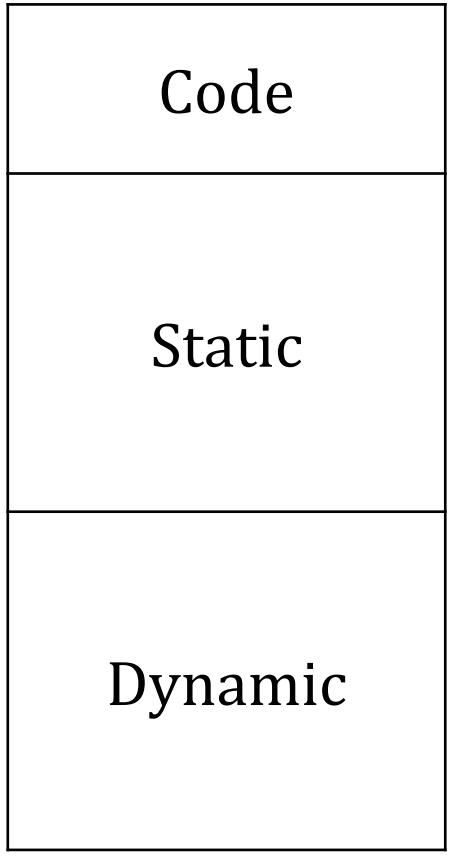
Storage Organization (Cont…)



- Runtime Memory is first subdivided into two areas: **Code** and **Data** Areas
- The size of the generated target code is fixed at compile time, so the compiler can place the executable target code in a statically determined area **Code** usually in the low end of memory
- Data area is further subdivided into two areas: **Static** and **Dynamic** areas

Typical Subdivision of Runtime Memory into Code and Data Areas

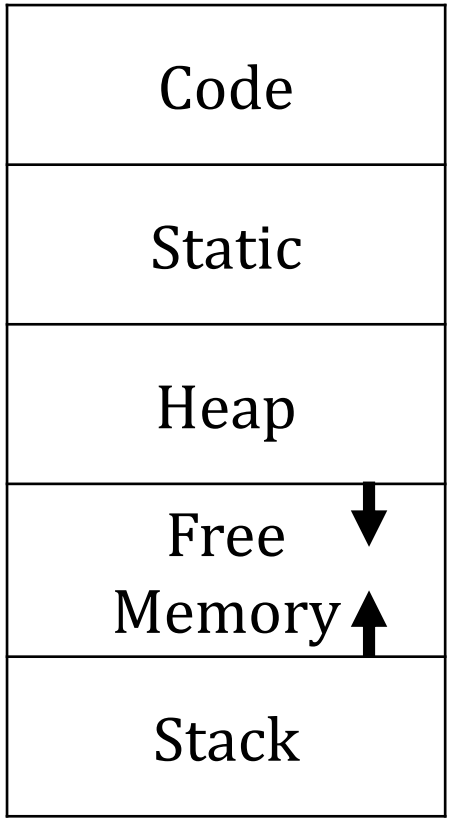
Storage Organization (Cont…)



- The size of some program data objects such as global constants and data generated by the compiler such as information to support garbage collection may be known at compile time and these data objects can be placed in another statically determined area called **Static**
- Dynamic Data area is further subdivided into two areas: **Stack** and **Heap** areas

Typical Subdivision of Runtime Memory into Code and Data Areas

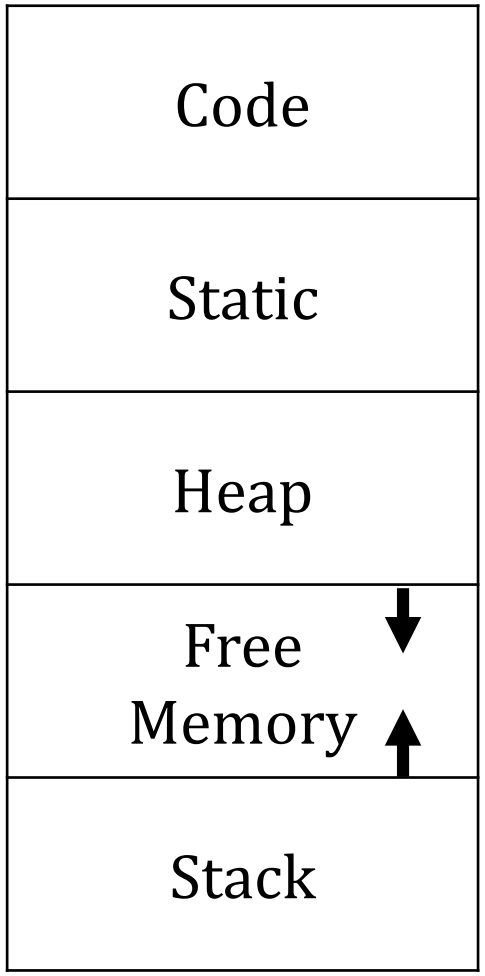
Storage Organization (Cont…)



- To maximize the utilization of space at run time the other two areas **Stack** and **Heap** are at the opposite ends of the remainder of the address space
- These areas are dynamic
- Their size can change as the program executes
- These areas grow towards each other as needed

Typical Subdivision of Runtime Memory into Code and Data Areas

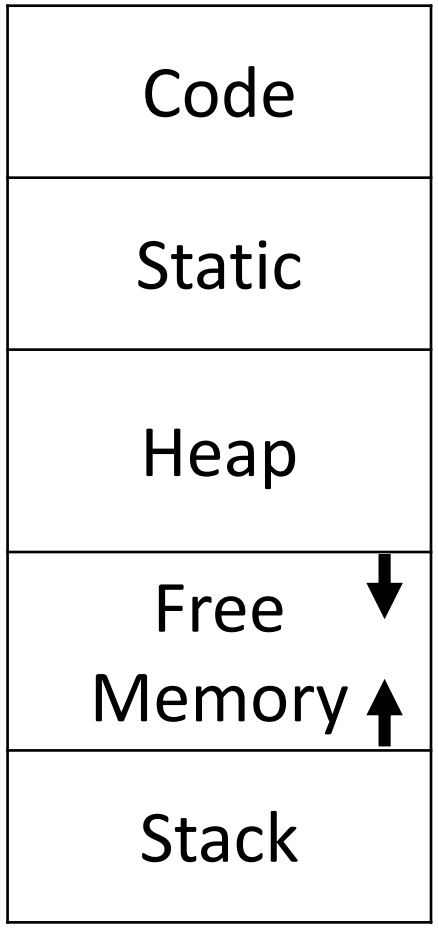
Storage Organization (Cont…)



- The **stack** is used to store data structures called activation records that get generated during procedure calls
- In practice, the **stack** grows towards lower addresses and the **heap** towards higher
- However, in this book the authors assume that the stack grows towards higher addresses so that they can use positive offsets for examples

Typical Subdivision of Runtime Memory into Code and Data Areas

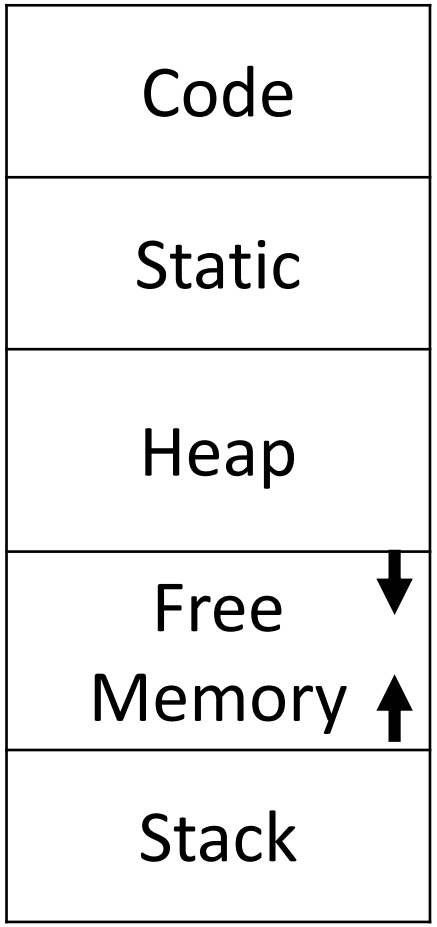
Storage Organization (Cont…)



- An activation record is used to store information about the status of the machine such as the value of the program counter and machine registers when a procedure call occurs
- When control returns from the call the activation of the calling procedure can be restarted after restoring the values of relevant registers and setting the program counter to the point immediately after the call
- Data objects whose lifetimes are contained in that of an activation can be allocated on the stack along with other information associated with the activation

Typical Subdivision of Runtime Memory into Code and Data Areas

Storage Organization (Cont…)



- Many programming languages allow the programmer to allocate and deallocate data under program control
- For example, C has the functions malloc and free that can be used to obtain and give back arbitrary chunks of storage
- The **heap** is used to manage this kind of dynamic data

Typical Subdivision of Runtime Memory into Code and Data Areas

Padding

- The storage layout for data objects is strongly influenced by the addressing constraints of the target machine
- On many machines instructions to add integers may expect integers to be aligned that is placed at an address divisible by 4
- Although an array of ten characters needs only enough bytes to hold ten characters, a compiler may allocate 12 bytes to get the proper alignment, leaving 2 bytes unused
- Space left unused due to alignment considerations is referred to as **padding**
- When space is at a premium, a compiler may pack data so that no padding is left. Additional instructions may then need to be executed at run time to position packed data so that it can be operated on as if it were properly aligned

Static Versus Dynamic Storage Allocation

- The layout and allocation of data to memory locations in the run-time environment are key issues in storage management
- The two adjectives static and dynamic distinguish between compile time and run time respectively
- We say that a storage-allocation decision is static if it can be made by the compiler looking only at the text of the program not at what the program does when it executes
- Conversely, a decision is dynamic if it can be decided only while the program is running

Static Versus Dynamic Storage Allocation (Cont…)

- Many compilers use some combination of the following two strategies for dynamic storage allocation:
  - **Stack storage:** Names local to a procedure are allocated space on a stack
  - **Heap storage:** Data that may outlive the call to the procedure that created it is usually allocated on a “heap” of reusable storage

Stack Allocation of Space

- Almost all compilers for languages that use procedures, functions or methods as units of user-defined actions manage at least part of their run-time memory as a stack
- Each time a procedure is called space for its local variables is pushed onto a stack and when the procedure terminates that space is popped off the stack
- This arrangement allows space to be shared by procedure calls whose durations do not overlap in time
- It allows to compile code for a procedure in such a way that the relative addresses of its nonlocal variables are always the same regardless of the sequence of procedure calls

Activation Trees

Stack allocation would not be feasible if procedure calls, or activations of procedures did not nest in time

Example 7.1

- Figure contains a sketch of a program that reads nine integers into an array a and sorts them using the recursive quicksort algorithm
- The main function has three tasks
- It calls readArray, sets the sentinels, and then calls quicksort on the entire data array

```
int a[11];
void readArray() { /* Reads 9 integers into a[1], ..., a[9]. */
    int i;
    ...
}
int partition(int m, int n) {
    /* Picks a separator value v, and partitions a[m..n] so that
       a[m..p-1] are less than v, a[p] = v, and a[p+1..n] are
       equal to or greater than v. Returns p. */
    ...
}
void quicksort(int m, int n) {
    int i;
    if (n > m) {
        i = partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
    }
}
main() {
    readArray();
    a[0] = -9999;
    a[10] = 9999;
    quicksort(1,9);
}
```

Sketch of a quicksort program

Example 7.1 (Cont…)

- This program uses two auxiliary functions readArray and partition
- The function readArray is used only to load the data into the array a
- The first and last elements of a are used for sentinels set in the main function
- We assume a[0] and a[10] are set to a value lower than and higher than any possible data value respectively

```
int a[11];
void readArray() { /* Reads 9 integers into a[1], ..., a[9]. */
    int i;
    ...
}
int partition(int m, int n) {
    /* Picks a separator value v, and partitions a[m..n] so that
       a[m..p-1] are less than v, a[p] = v, and a[p+1..n] are
       equal to or greater than v. Returns p. */
    ...
}
void quicksort(int m, int n) {
    int i;
    if (n > m) {
        i = partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
    }
}
main() {
    readArray();
    a[0] = -9999;
    a[10] = 9999;
    quicksort(1,9);
}
```

Sketch of a quicksort program

Example 7.1 (Cont...)

- The partition function divides a portion of the array, delimited by the arguments m and n
- So the low elements of a[m] through a[n] are at the beginning and the high elements are at the end
- Although neither group is necessarily in the sorted order

```
int a[11];
void readArray() { /* Reads 9 integers into a[1], ..., a[9]. */
    int i;
    ...
}
int partition(int m, int n) {
    /* Picks a separator value v, and partitions a[m..n] so that
       a[m..p-1] are less than v, a[p] = v, and a[p+1..n] are
       equal to or greater than v. Returns p. */
    ...
}
void quicksort(int m, int n) {
    int i;
    if (n > m) {
        i = partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
    }
}
main() {
    readArray();
    a[0] = -9999;
    a[10] = 9999;
    quicksort(1,9);
}
```

Sketch of a quicksort program

Example 7.1 (Cont...)

- Quicksort procedure check if the condition n>m is true or not and if it is true then it calls partition function otherwise return
- Partition function returns an index i to separate the low and high elements
- These two groups of elements are then sorted by two recursive calls to quicksort

```
int a[11];
void readArray() { /* Reads 9 integers into a[1], ..., a[9]. */
    int i;
    ...
}
int partition(int m, int n) {
    /* Picks a separator value v, and partitions a[m..n] so that
       a[m..p-1] are less than v, a[p] = v, and a[p+1..n] are
       equal to or greater than v. Returns p. */
    ...
}
void quicksort(int m, int n) {
    int i;
    if (n > m) {
        i = partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
    }
}
main() {
    readArray();
    a[0] = -9999;
    a[10] = 9999;
    quicksort(1,9);
}
```

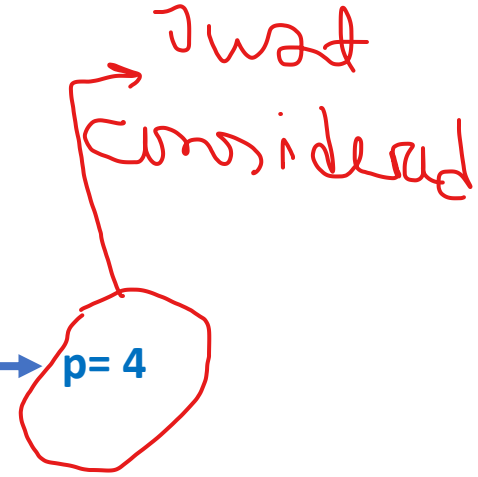
Sketch of a quicksort program

Example 7.1 (Cont...)

```
int a[11];
void readArray() { /* Reads 9 integers into a[1], ..., a[9]. */
    int i;
    ...
}
int partition(int m, int n) {
    /* Picks a separator value v, and partitions a[m..n] so that
       a[m..p-1] are less than v, a[p] = v, and a[p+1..n] are
       equal to or greater than v. Returns p. */
    ...
}
void quicksort(int m, int n) {
    int i;
    if (n > m) {
        i = partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
    }
}
main() {
    readArray();
    a[0] = -9999;
    a[10] = 9999;
    quicksort(1,9);
}
```

Sketch of a quicksort program

enter main ()
  enter readArray ()
  leave readArray ()
  enter quicksort (1,9)
    enter partition (1,9) → p= 4
    leave partition (1,9)
    enter quicksort (1,3)
      . . .
    leave quicksort (1,3)
    enter quicksort (5,9)
      . . .
    leave quicksort (5,9)
  leave quicksort (1,9)
leave main ()



Possible Activations for the Quicksort Program

Example 7.1 (Cont…)

- Figure suggests a sequence of calls that might result from an execution of the program
- In this execution, the call to partition(1,9) returns 4, so a[1] through a[3] hold elements less than its chosen separator value v, while the larger elements are in a[5] through a[9]

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort  (1,9)
        enter partition (1,9)  → p= 4
        leave partition (1,9)
        enter quicksort (1,3)
            . . .
        leave quicksort (1,3)
        enter quicksort (5,9)
            . . .
        leave quicksort (5,9)
    leave quicksort (1,9)
leave main ()
```

Possible Activations for the Quicksort Program

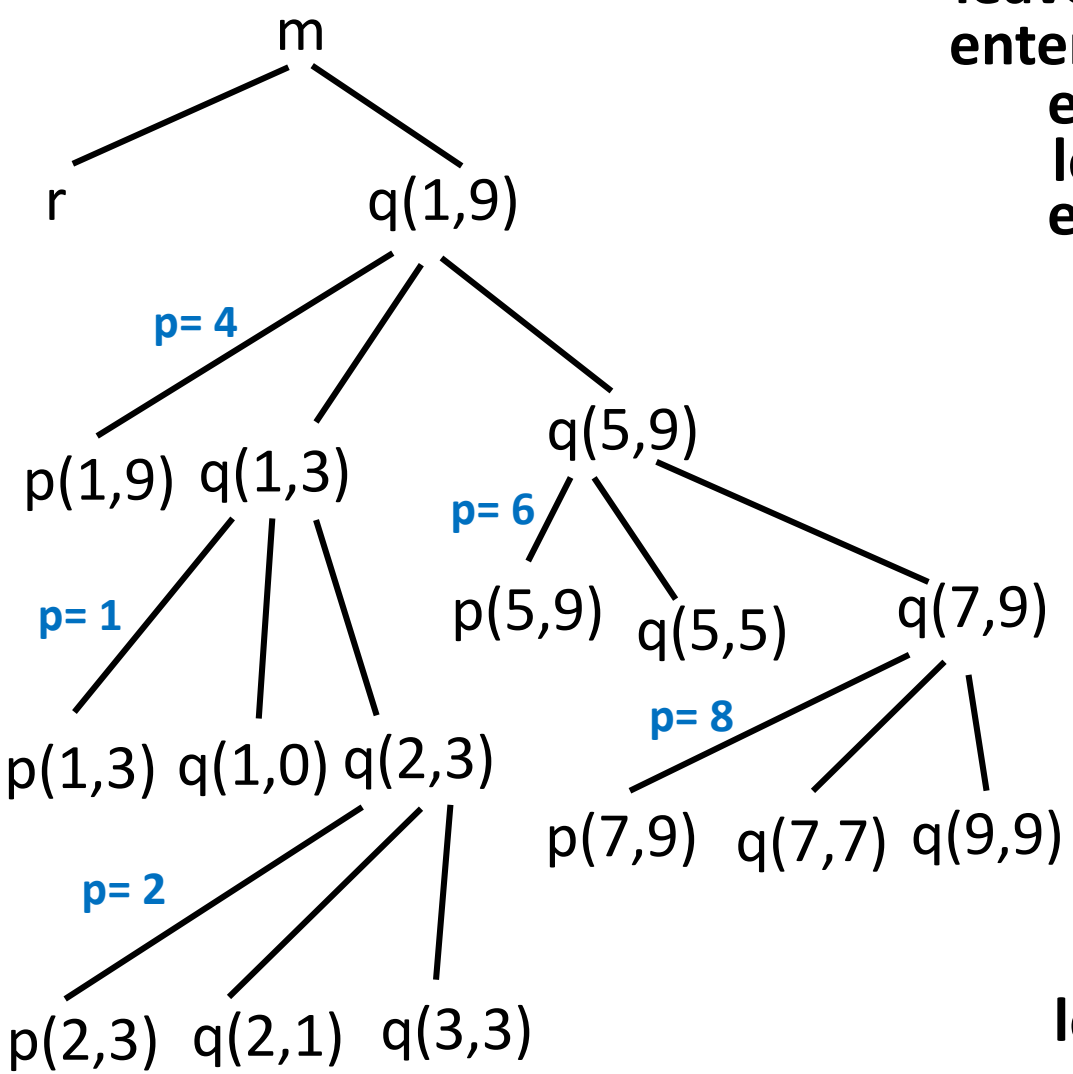
Activation Trees (Cont…)

- In this example, as is true in general procedure activations are nested in time
- If an activation of procedure p calls procedure q then that activation of q must end before the activation of p can end
- There are three common cases:
  - **The activation of q terminates normally**
  - **The activation of q or some procedure q called either directly or indirectly aborts**
  - **The activation of q terminates because of an exception that q cannot handle**

Activation Trees (Cont…)

- We therefore can represent the activations of procedures during the running of an entire program by a tree, called an activation tree
- Each node corresponds to one activation and the root is the activation of the “main” procedure that initiates execution of the program
- At a node for an activation of procedure p the children correspond to activations of the procedures called by this activation of p
- We show these activations in the order that they are called from left to right
- Notice that one child must finish before the activation to its right can begin

Example 7.2



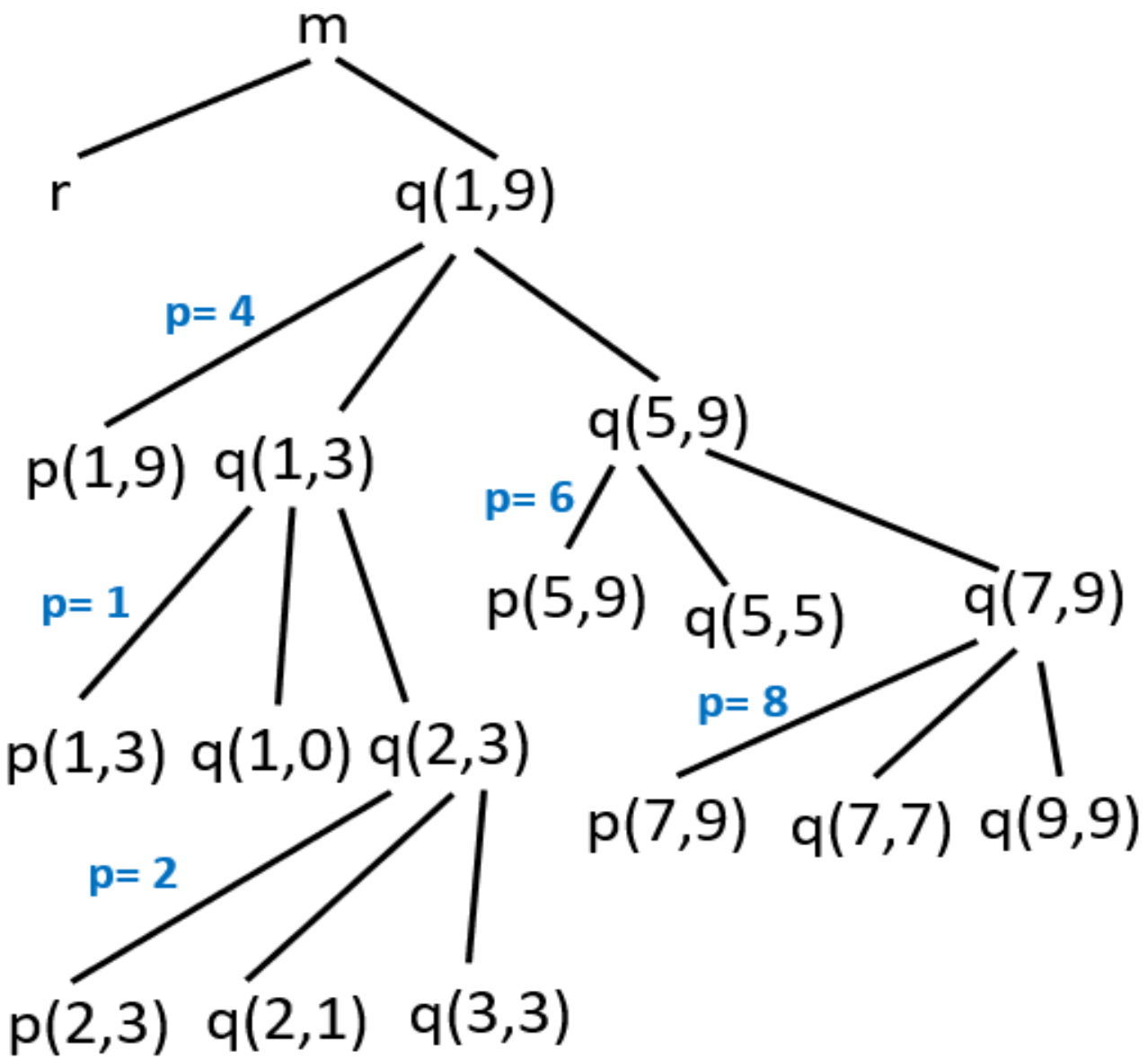
Activation Tree Representing Calls During an Execution of Quicksort

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort  (1,9)
        enter partition (1,9)  → p= 4
        leave partition (1,9)
        enter quicksort (1,3)
            enter partition (1,3)  → p= 1
            leave partition (1,3)
            enter quicksort (1,0)
            leave quicksort (1,0)
            enter quicksort (2,3)
                enter partition (2,3)  → p= 2
                leave partition  (2,3)
                enter quicksort (2,1)
                leave quicksort (2,1)
                enter quicksort (3,3)
                leave quicksort (3,3)
            leave quicksort (2,3)
        leave quicksort (1,3)
        enter quicksort (5,9)
            enter partition (5,9)  → p= 6
            leave partition (5,9)
            enter quicksort (5,5)
            leave quicksort (5,5)
            enter quicksort (7,9)  p= 8
                enter partition (7,9)
                leave partition  (7,9)
                enter quicksort (7,7)
                leave quicksort (7,7)
                enter quicksort (9,9)
                leave quicksort (9,9)
            leave quicksort (7,9)
        leave quicksort (5,9)
    leave quicksort (1,9)
leave main ()
```

Possible Activations for the Quicksort Program

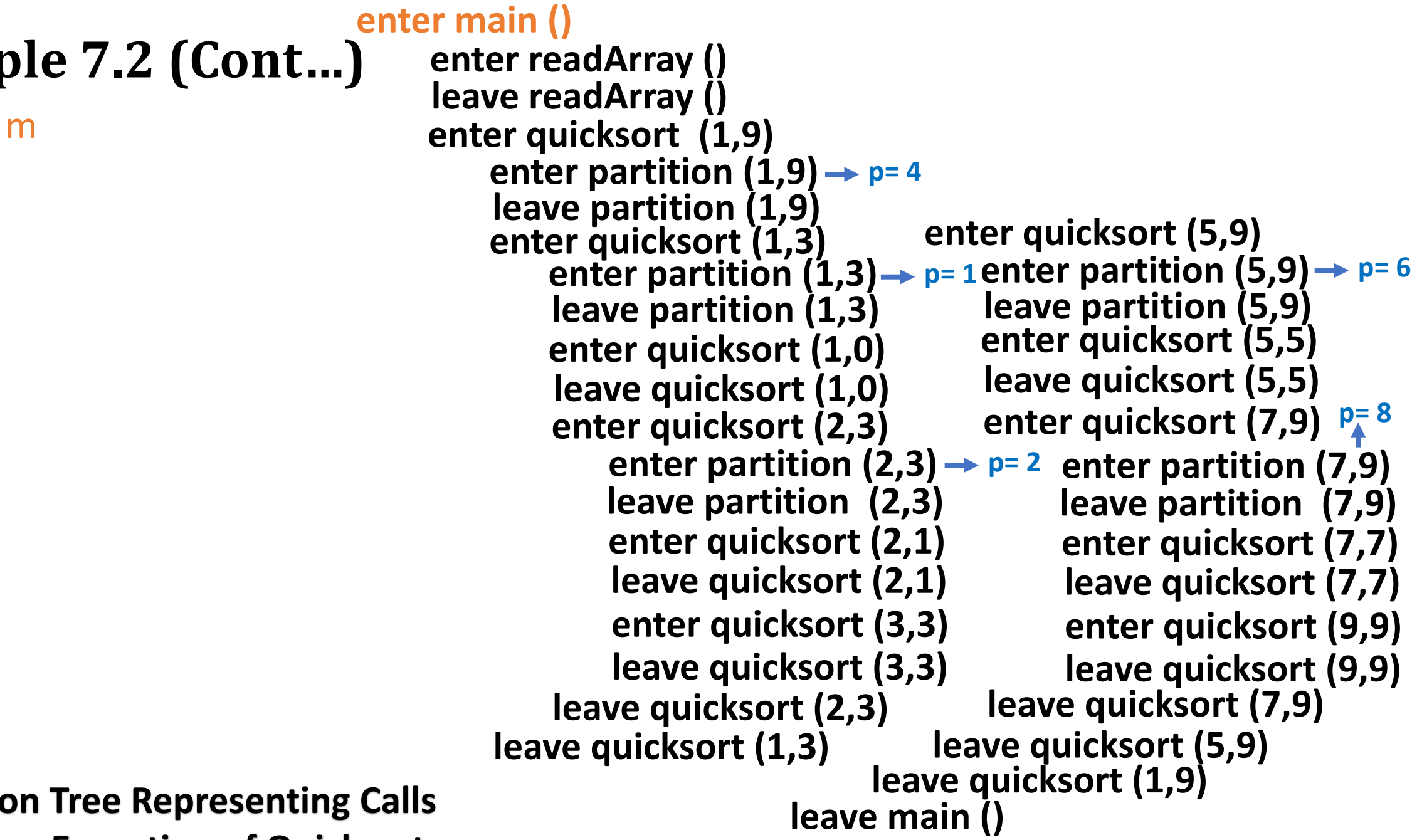
Example 7.2(Cont…)

- One possible activation tree that completes the sequence of calls and returns suggested is shown in figure
- Functions are represented by the first letters of their names
- Remember that this tree is only one possibility since the arguments of subsequent calls and also the number of calls along any branch is influenced by the values returned by partition



Activation Tree Representing Calls During an Execution of Quicksort

Example 7.2 (Cont…)

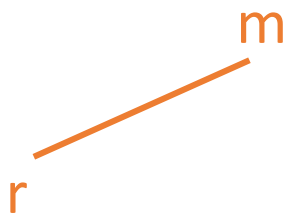


m

Activation Tree Representing Calls During an Execution of Quicksort

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

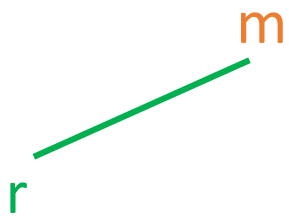


Activation Tree Representing Calls During an Execution of Quicksort

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort  (1,9)
        enter partition (1,9) → p= 4
        leave partition (1,9)
        enter quicksort (1,3)                    enter quicksort (5,9)
            enter partition (1,3) → p= 1          enter partition (5,9) → p= 6
            leave partition (1,3)                  leave partition (5,9)
            enter quicksort (1,0)                  enter quicksort (5,5)
            leave quicksort (1,0)                  leave quicksort (5,5)
            enter quicksort (2,3)                  enter quicksort (7,9)  p= 8
                enter partition (2,3) → p= 2       enter partition (7,9)
                leave partition  (2,3)             leave partition  (7,9)
                enter quicksort (2,1)              enter quicksort (7,7)
                leave quicksort (2,1)              leave quicksort (7,7)
                enter quicksort (3,3)              enter quicksort (9,9)
                leave quicksort (3,3)              leave quicksort (9,9)
            leave quicksort (2,3)                  leave quicksort (7,9)
        leave quicksort (1,3)                      leave quicksort (5,9)
                                                   leave quicksort (1,9)
                                                   leave main ()
```

Possible Activations for the Quicksort Program

Example 7.2 (Cont...)

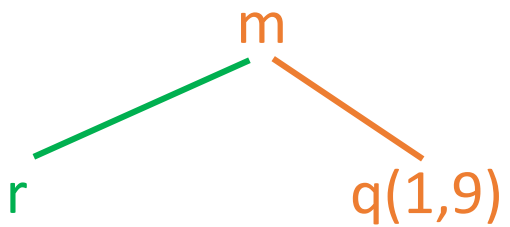


Activation Tree Representing Calls During an Execution of Quicksort

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort (1,9)
        enter partition (1,9) → p= 4
        leave partition (1,9)
        enter quicksort (1,3)                    enter quicksort (5,9)
            enter partition (1,3) → p= 1          enter partition (5,9) → p= 6
            leave partition (1,3)                  leave partition (5,9)
            enter quicksort (1,0)                  enter quicksort (5,5)
            leave quicksort (1,0)                  leave quicksort (5,5)
            enter quicksort (2,3)                  enter quicksort (7,9)  p= 8
                enter partition (2,3) → p= 2       enter partition (7,9)
                leave partition (2,3)              leave partition (7,9)
                enter quicksort (2,1)              enter quicksort (7,7)
                leave quicksort (2,1)              leave quicksort (7,7)
                enter quicksort (3,3)              enter quicksort (9,9)
                leave quicksort (3,3)              leave quicksort (9,9)
            leave quicksort (2,3)                  leave quicksort (7,9)
        leave quicksort (1,3)                  leave quicksort (5,9)
                                           leave quicksort (1,9)
                                       leave main ()
```

Possible Activations for the Quicksort Program

Example 7.2 (Cont...)

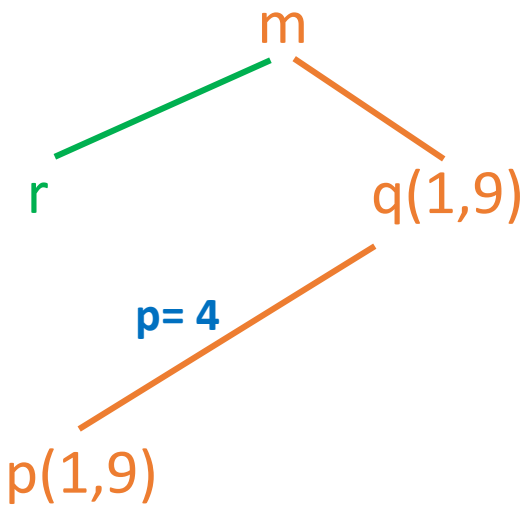


Activation Tree Representing Calls During an Execution of Quicksort

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort  (1,9)
        enter partition (1,9)  → p= 4
        leave partition (1,9)
        enter quicksort (1,3)                    enter quicksort (5,9)
            enter partition (1,3) → p= 1          enter partition (5,9) → p= 6
            leave partition (1,3)                  leave partition (5,9)
            enter quicksort (1,0)                  enter quicksort (5,5)
            leave quicksort (1,0)                  leave quicksort (5,5)
            enter quicksort (2,3)                  enter quicksort (7,9)  p= 8
                enter partition (2,3) → p= 2        enter partition (7,9)
                leave partition  (2,3)              leave partition  (7,9)
                enter quicksort (2,1)               enter quicksort (7,7)
                leave quicksort (2,1)               leave quicksort (7,7)
                enter quicksort (3,3)               enter quicksort (9,9)
                leave quicksort (3,3)               leave quicksort (9,9)
            leave quicksort (2,3)                  leave quicksort (7,9)
        leave quicksort (1,3)                    leave quicksort (5,9)
                                                leave quicksort (1,9)
                                            leave main ()
```

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

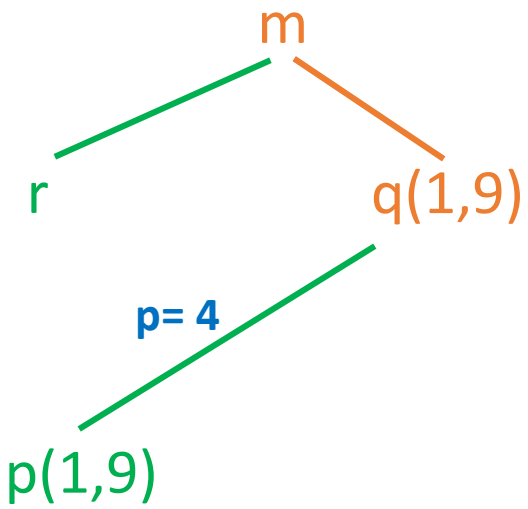


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

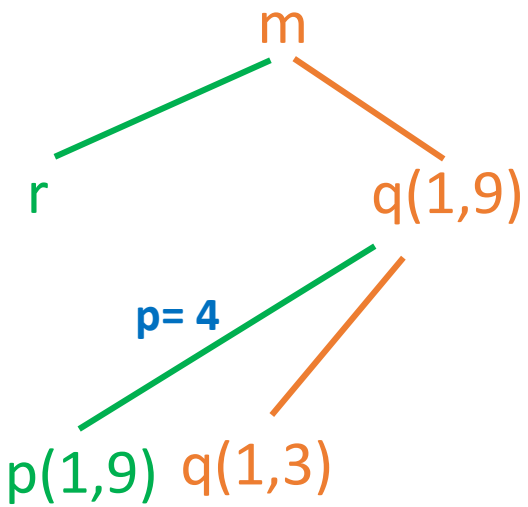


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

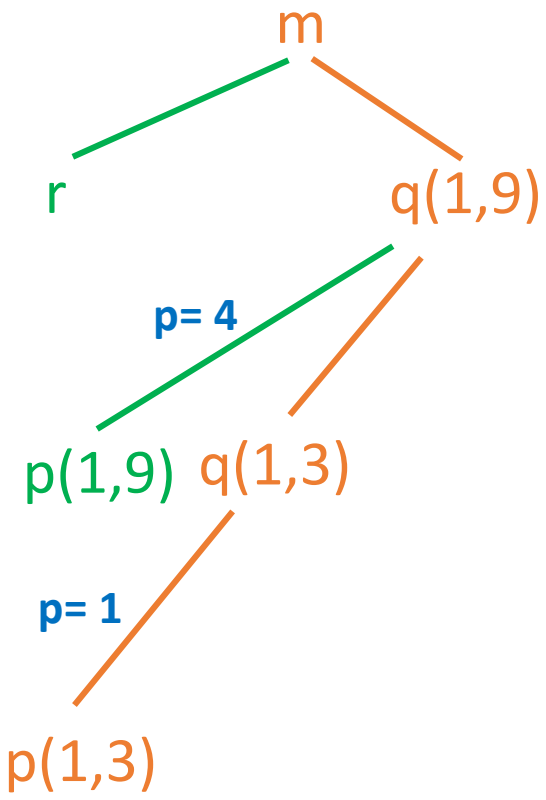


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

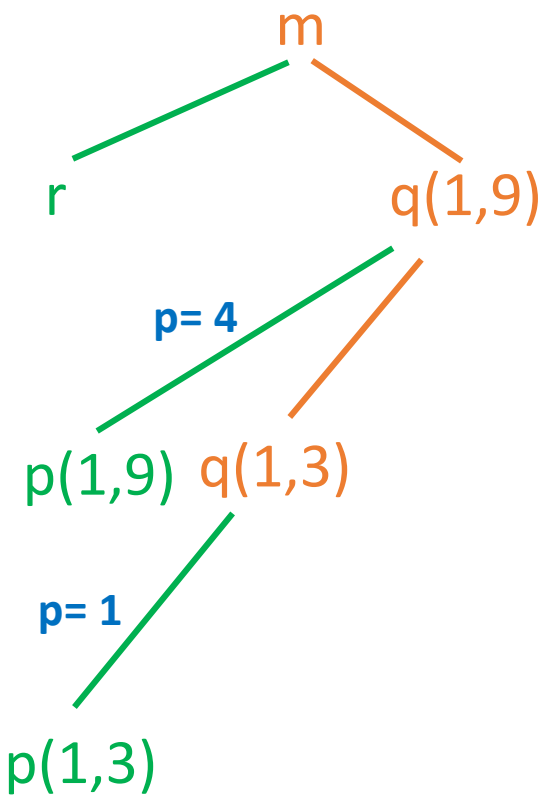


Activation Tree Representing Calls During an Execution of Quicksort

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort  (1,9)
        enter partition (1,9) → p= 4
        leave partition (1,9)
        enter quicksort (1,3)
            enter partition (1,3) → p= 1
            leave partition (1,3)
            enter quicksort (1,0)
            leave quicksort (1,0)
            enter quicksort (2,3)
                enter partition (2,3) → p= 2
                leave partition  (2,3)
                enter quicksort (2,1)
                leave quicksort (2,1)
                enter quicksort (3,3)
                leave quicksort (3,3)
            leave quicksort (2,3)
        leave quicksort (1,3)
        enter quicksort (5,9)
            enter partition (5,9) → p= 6
            leave partition (5,9)
            enter quicksort (5,5)
            leave quicksort (5,5)
            enter quicksort (7,9)   p= 8
                enter partition (7,9)
                leave partition  (7,9)
                enter quicksort (7,7)
                leave quicksort (7,7)
                enter quicksort (9,9)
                leave quicksort (9,9)
            leave quicksort (7,9)
        leave quicksort (5,9)
    leave quicksort (1,9)
leave main ()
```

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

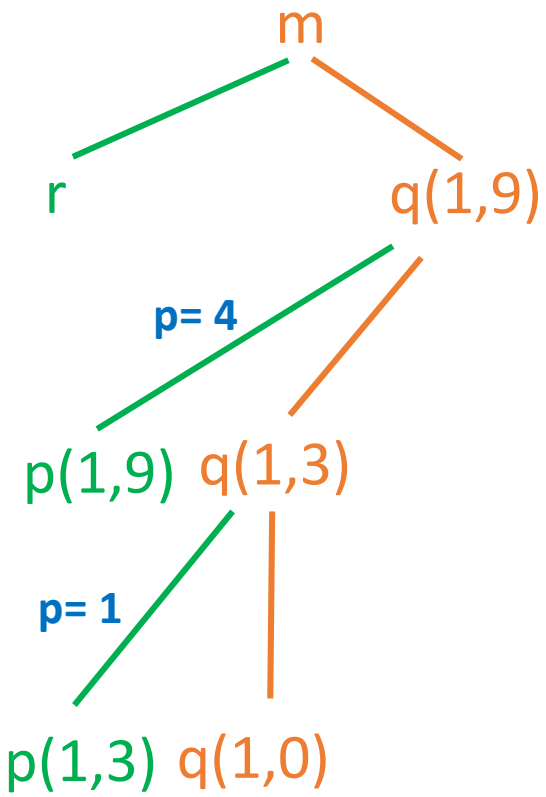


Activation Tree Representing Calls During an Execution of Quicksort

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort  (1,9)
        enter partition (1,9) → p= 4
        leave partition (1,9)
        enter quicksort (1,3)
            enter partition (1,3) → p= 1
            leave partition (1,3)
            enter quicksort (1,0)
            leave quicksort (1,0)
            enter quicksort (2,3)
                enter partition (2,3) → p= 2
                leave partition  (2,3)
                enter quicksort (2,1)
                leave quicksort (2,1)
                enter quicksort (3,3)
                leave quicksort (3,3)
            leave quicksort (2,3)
        leave quicksort (1,3)
        enter quicksort (5,9)
            enter partition (5,9) → p= 6
            leave partition (5,9)
            enter quicksort (5,5)
            leave quicksort (5,5)
            enter quicksort (7,9)  p= 8
                enter partition (7,9)
                leave partition  (7,9)
                enter quicksort (7,7)
                leave quicksort (7,7)
                enter quicksort (9,9)
                leave quicksort (9,9)
            leave quicksort (7,9)
        leave quicksort (5,9)
    leave quicksort (1,9)
leave main ()
```

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

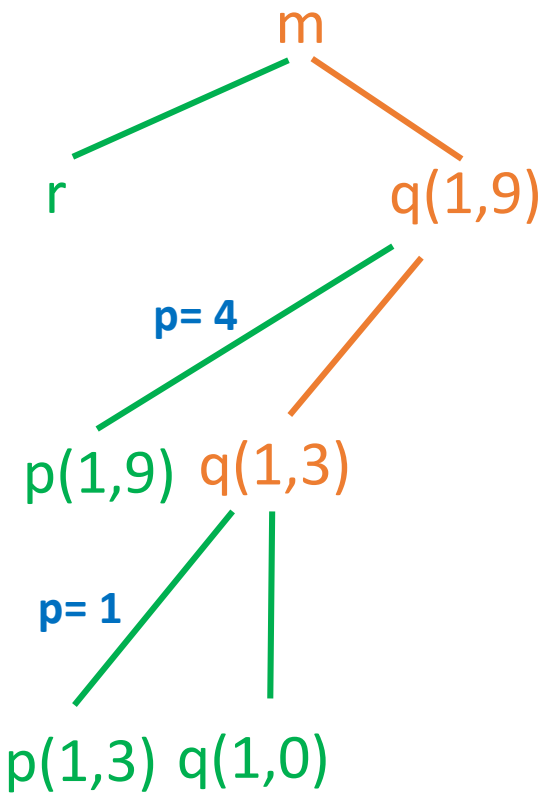


Activation Tree Representing Calls During an Execution of Quicksort

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort  (1,9)
        enter partition (1,9) → p= 4
        leave partition (1,9)
        enter quicksort (1,3)
            enter partition (1,3) → p= 1
            leave partition (1,3)
            enter quicksort (1,0)
            leave quicksort (1,0)
            enter quicksort (2,3)
                enter partition (2,3) → p= 2
                leave partition  (2,3)
                enter quicksort (2,1)
                leave quicksort (2,1)
                enter quicksort (3,3)
                leave quicksort (3,3)
            leave quicksort (2,3)
        leave quicksort (1,3)
        enter quicksort (5,9)
            enter partition (5,9) → p= 6
            leave partition (5,9)
            enter quicksort (5,5)
            leave quicksort (5,5)
            enter quicksort (7,9)   p= 8
                enter partition (7,9)
                leave partition  (7,9)
                enter quicksort (7,7)
                leave quicksort (7,7)
                enter quicksort (9,9)
                leave quicksort (9,9)
            leave quicksort (7,9)
        leave quicksort (5,9)
    leave quicksort (1,9)
leave main ()
```

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

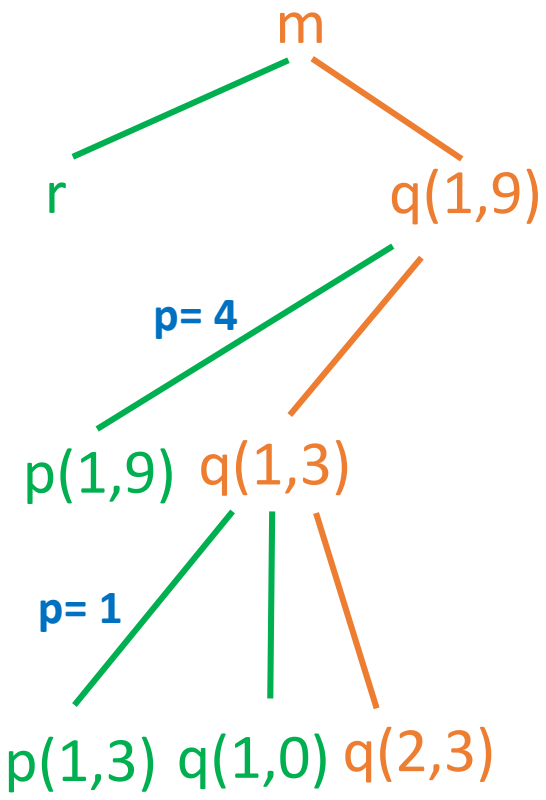


Activation Tree Representing Calls During an Execution of Quicksort

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort  (1,9)
        enter partition (1,9) → p= 4
        leave partition (1,9)
        enter quicksort (1,3)
            enter partition (1,3) → p= 1
            leave partition (1,3)
            enter quicksort (1,0)
            leave quicksort (1,0)
            enter quicksort (2,3)
                enter partition (2,3) → p= 2
                leave partition  (2,3)
                enter quicksort (2,1)
                leave quicksort (2,1)
                enter quicksort (3,3)
                leave quicksort (3,3)
            leave quicksort (2,3)
        leave quicksort (1,3)
        enter quicksort (5,9)
            enter partition (5,9) → p= 6
            leave partition (5,9)
            enter quicksort (5,5)
            leave quicksort (5,5)
            enter quicksort (7,9)  p= 8
                enter partition (7,9)
                leave partition  (7,9)
                enter quicksort (7,7)
                leave quicksort (7,7)
                enter quicksort (9,9)
                leave quicksort (9,9)
            leave quicksort (7,9)
        leave quicksort (5,9)
    leave quicksort (1,9)
leave main ()
```

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

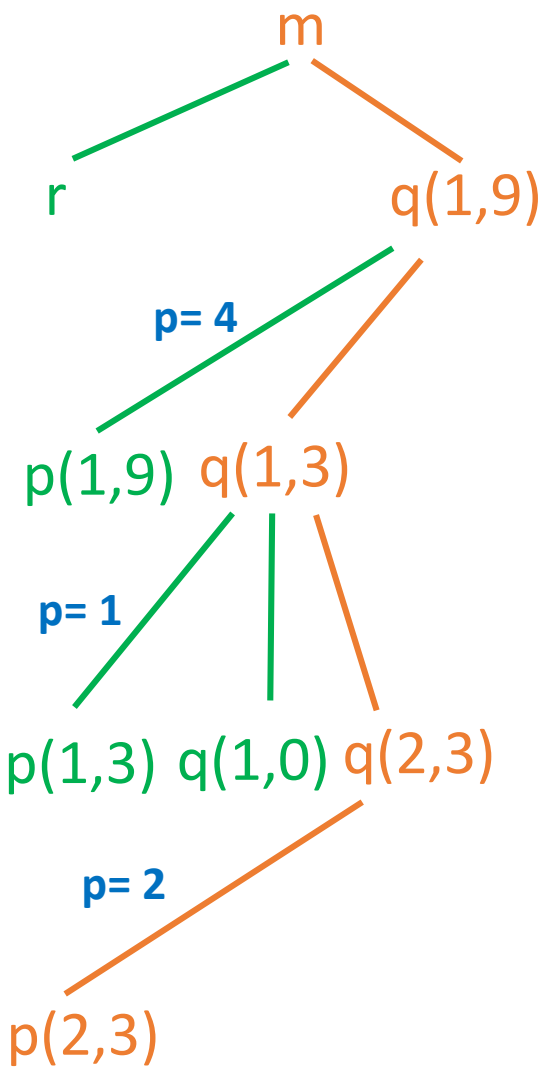


Activation Tree Representing Calls During an Execution of Quicksort

```
enter main ()
    enter readArray ()
    leave readArray ()
    enter quicksort  (1,9)
        enter partition (1,9) → p= 4
        leave partition (1,9)
        enter quicksort (1,3)
            enter partition (1,3) → p= 1
            leave partition (1,3)
            enter quicksort (1,0)
            leave quicksort (1,0)
            enter quicksort (2,3)
                enter partition (2,3) → p= 2
                leave partition  (2,3)
                enter quicksort (2,1)
                leave quicksort (2,1)
                enter quicksort (3,3)
                leave quicksort (3,3)
            leave quicksort (2,3)
        leave quicksort (1,3)
        enter quicksort (5,9)
            enter partition (5,9) → p= 6
            leave partition (5,9)
            enter quicksort (5,5)
            leave quicksort (5,5)
            enter quicksort (7,9)  p= 8
                enter partition (7,9)
                leave partition  (7,9)
                enter quicksort (7,7)
                leave quicksort (7,7)
                enter quicksort (9,9)
                leave quicksort (9,9)
            leave quicksort (7,9)
        leave quicksort (5,9)
    leave quicksort (1,9)
leave main ()
```

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

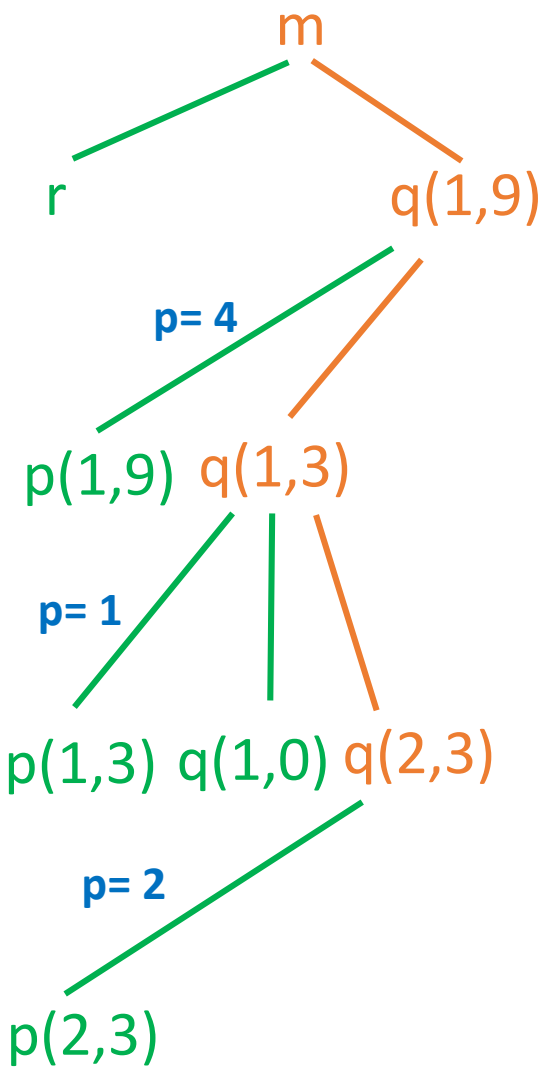


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

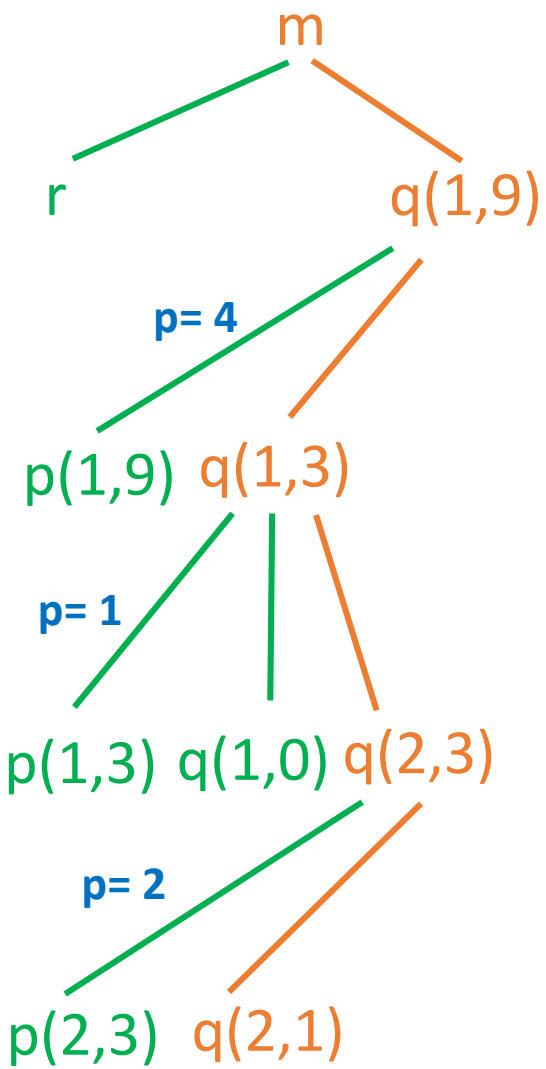


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

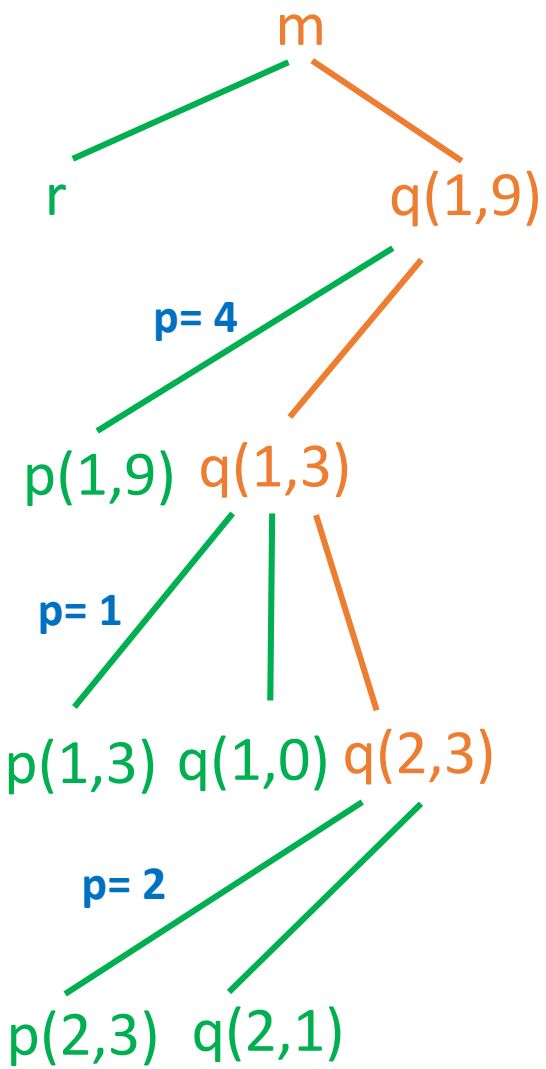


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

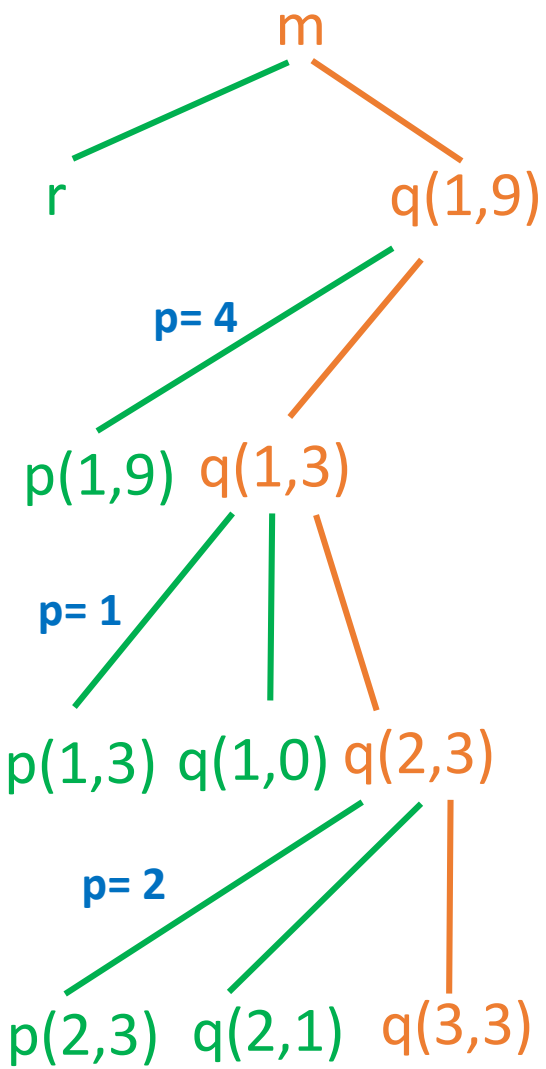


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

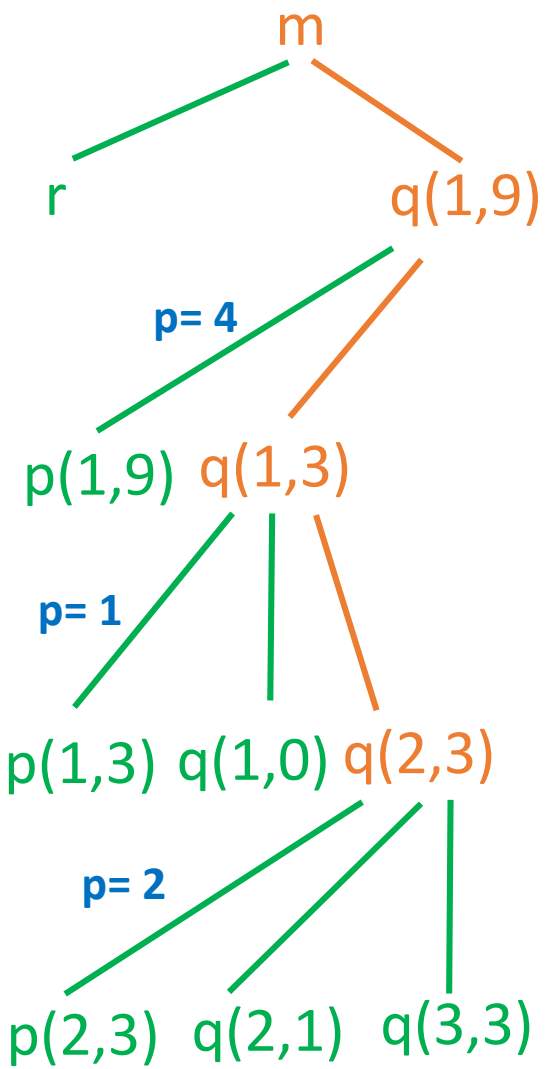


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

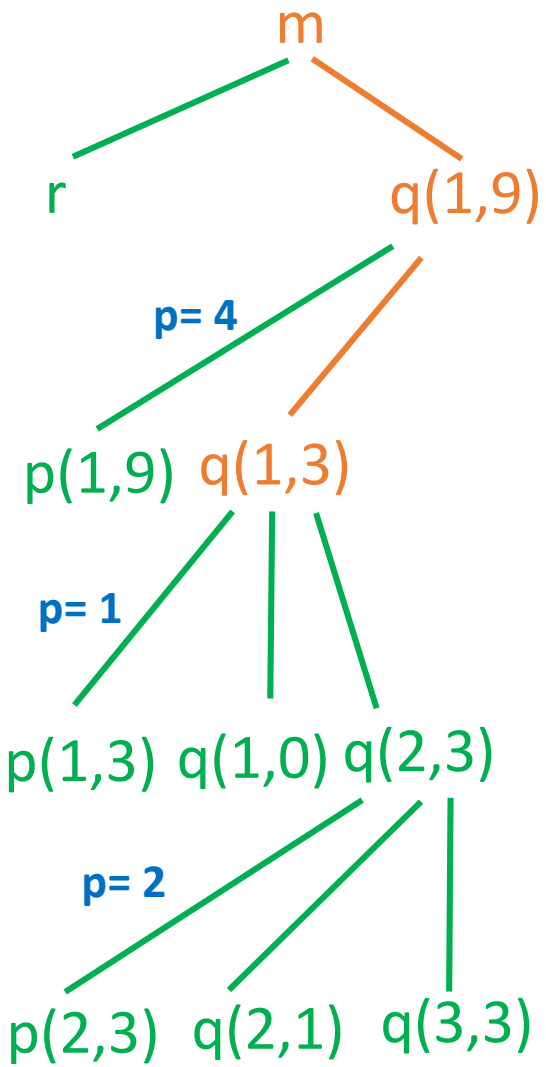


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

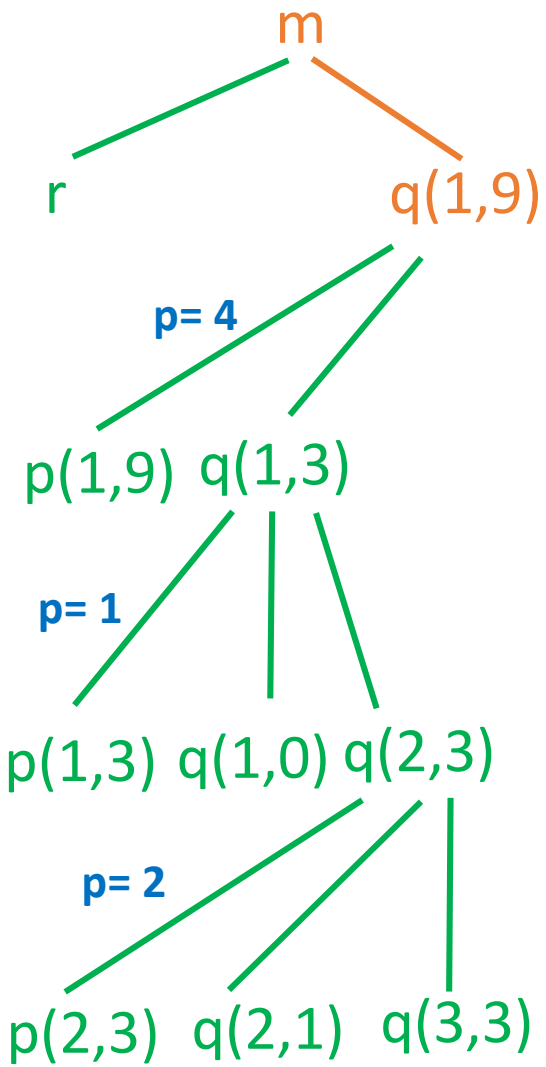


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

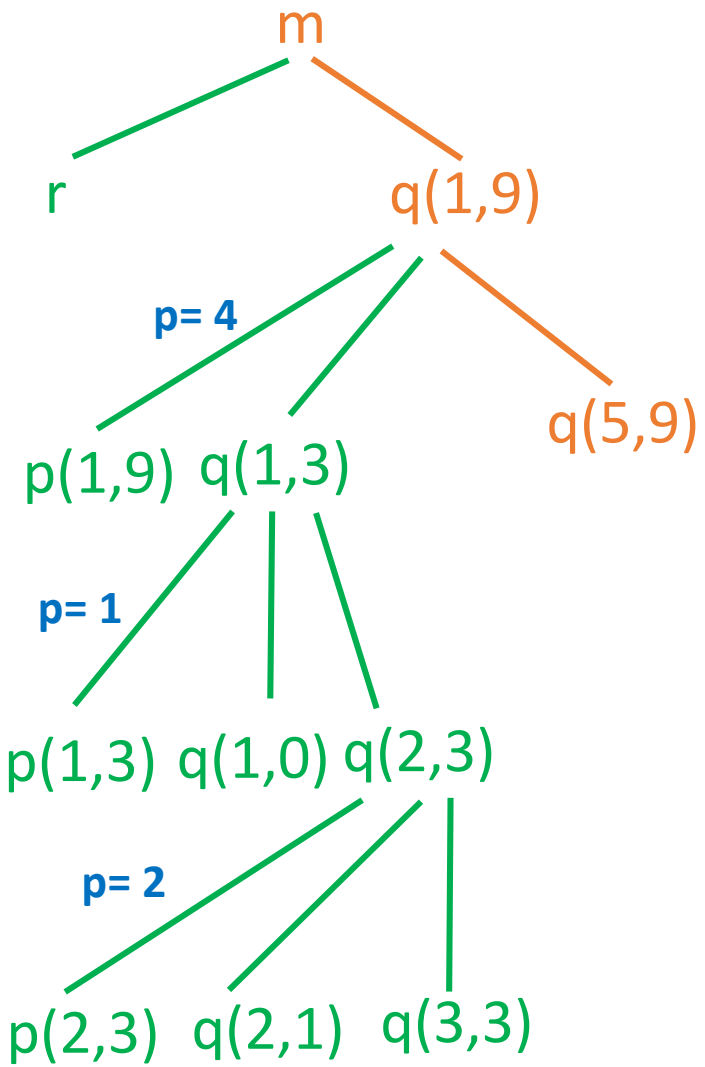


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

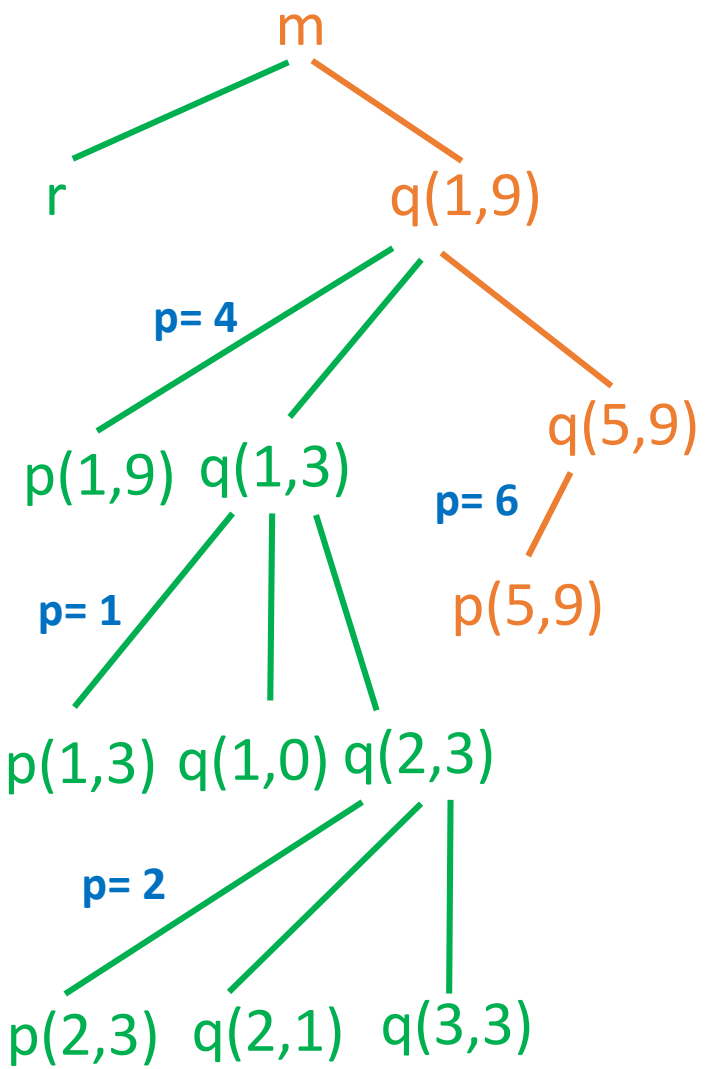


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)



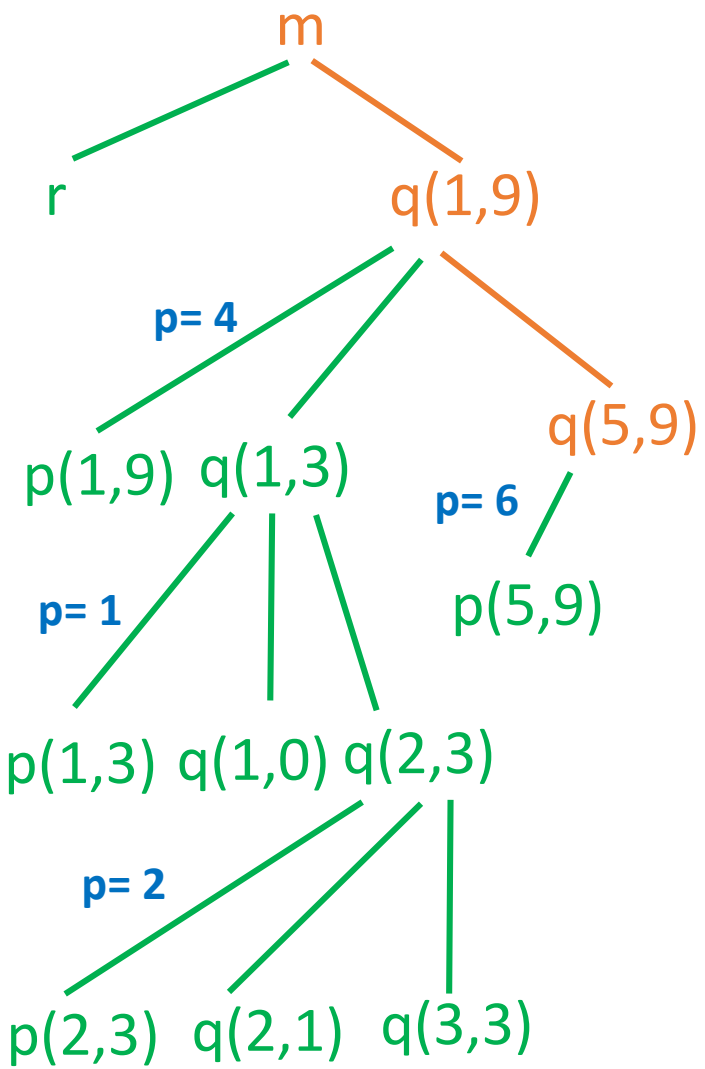
Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)

enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

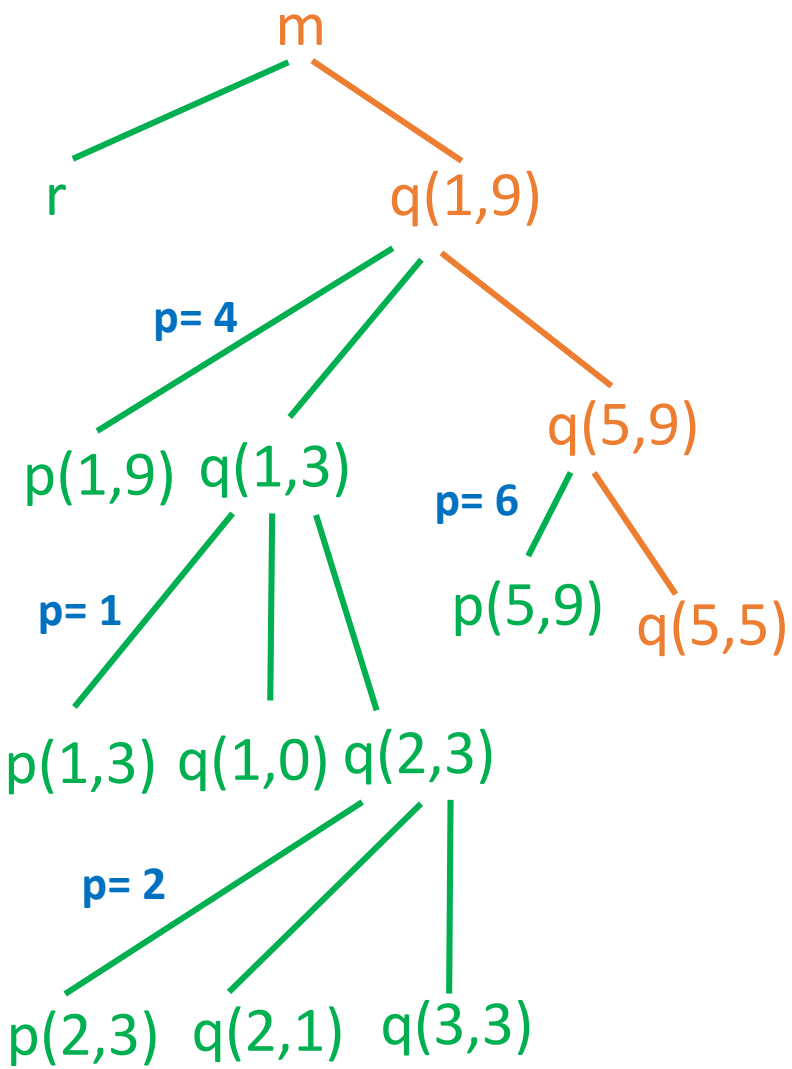


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

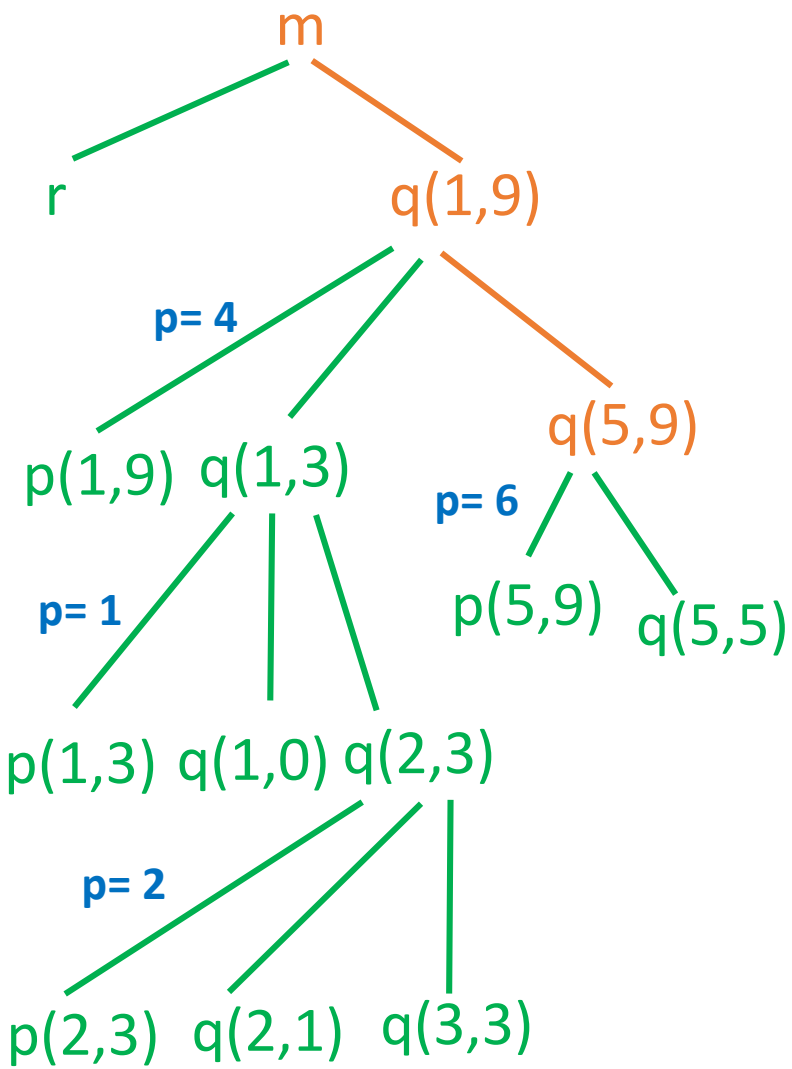


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

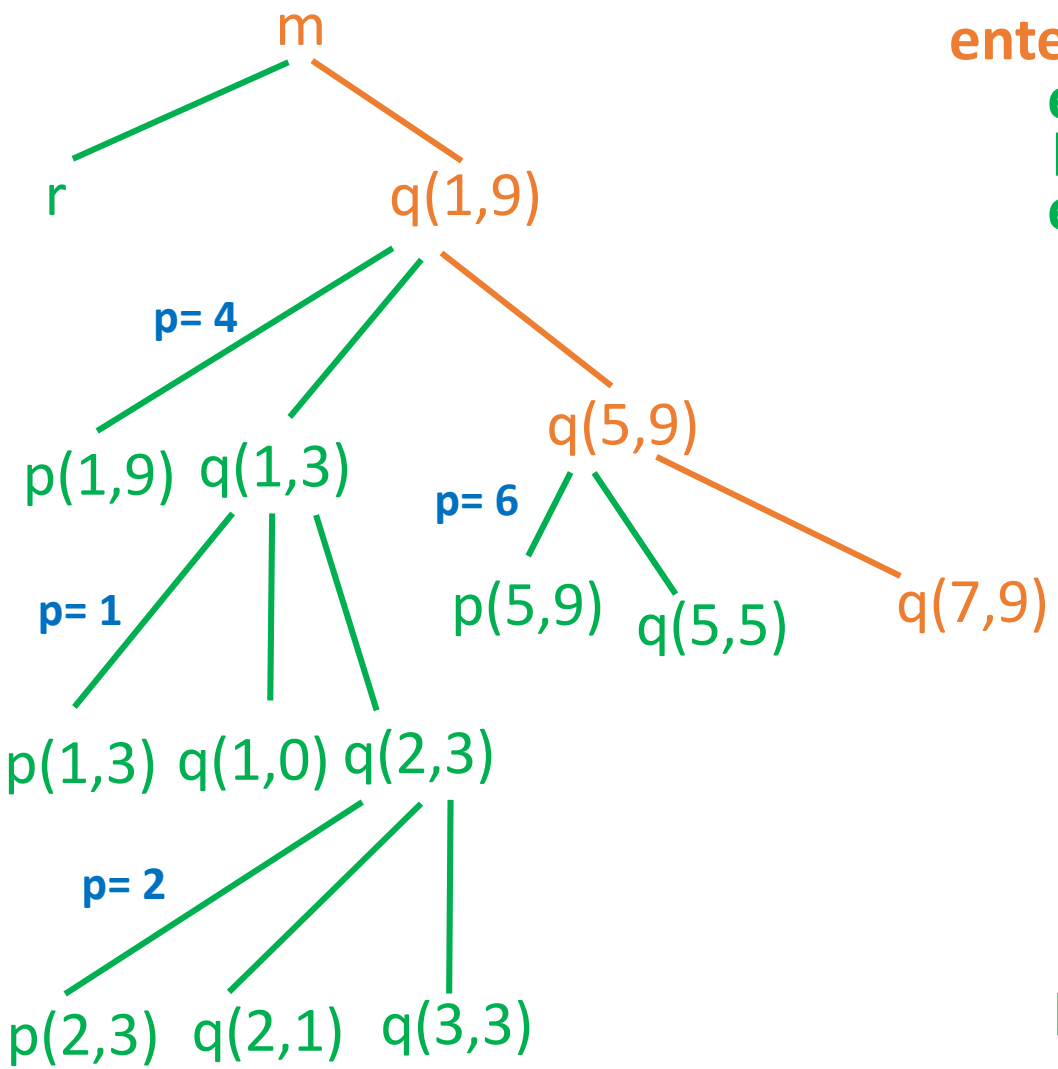


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

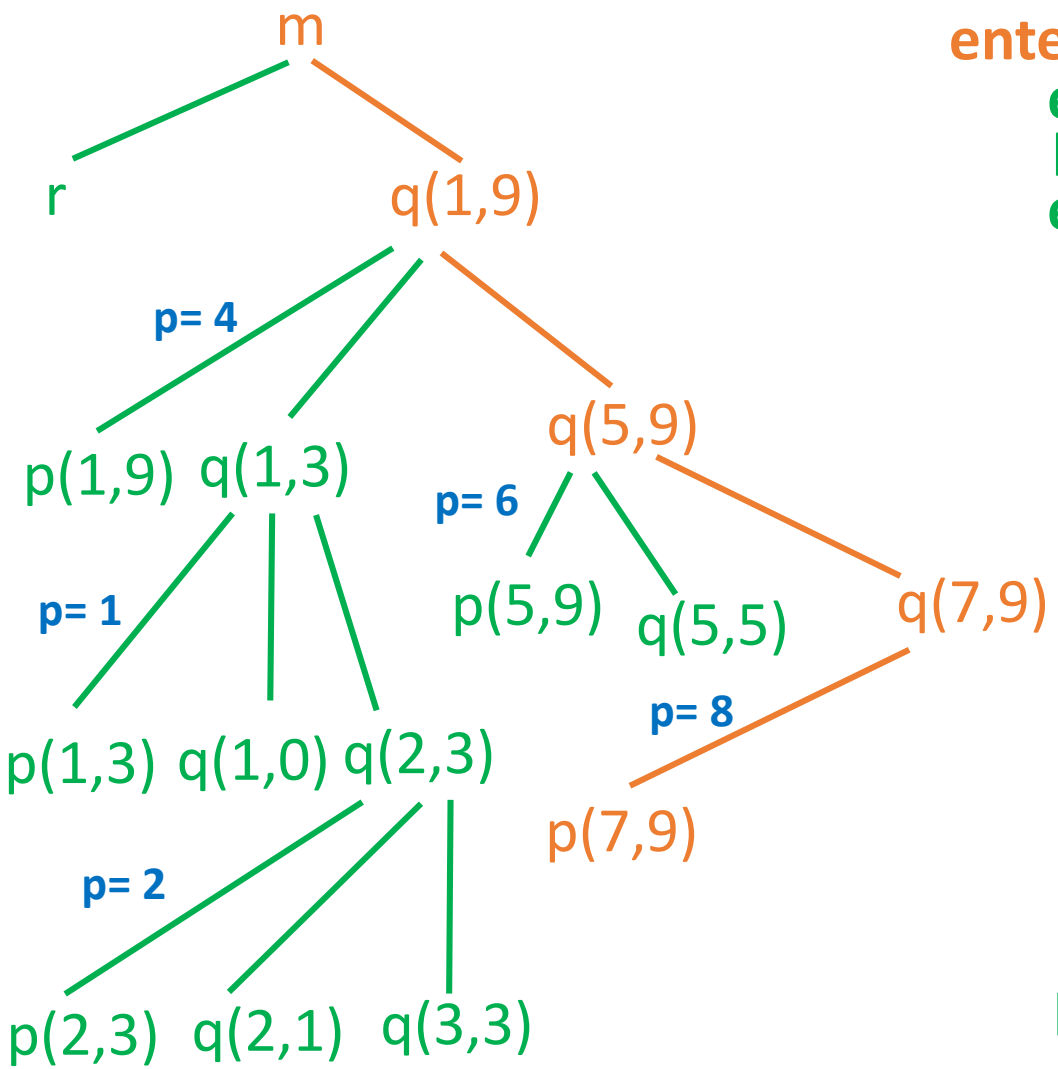


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)



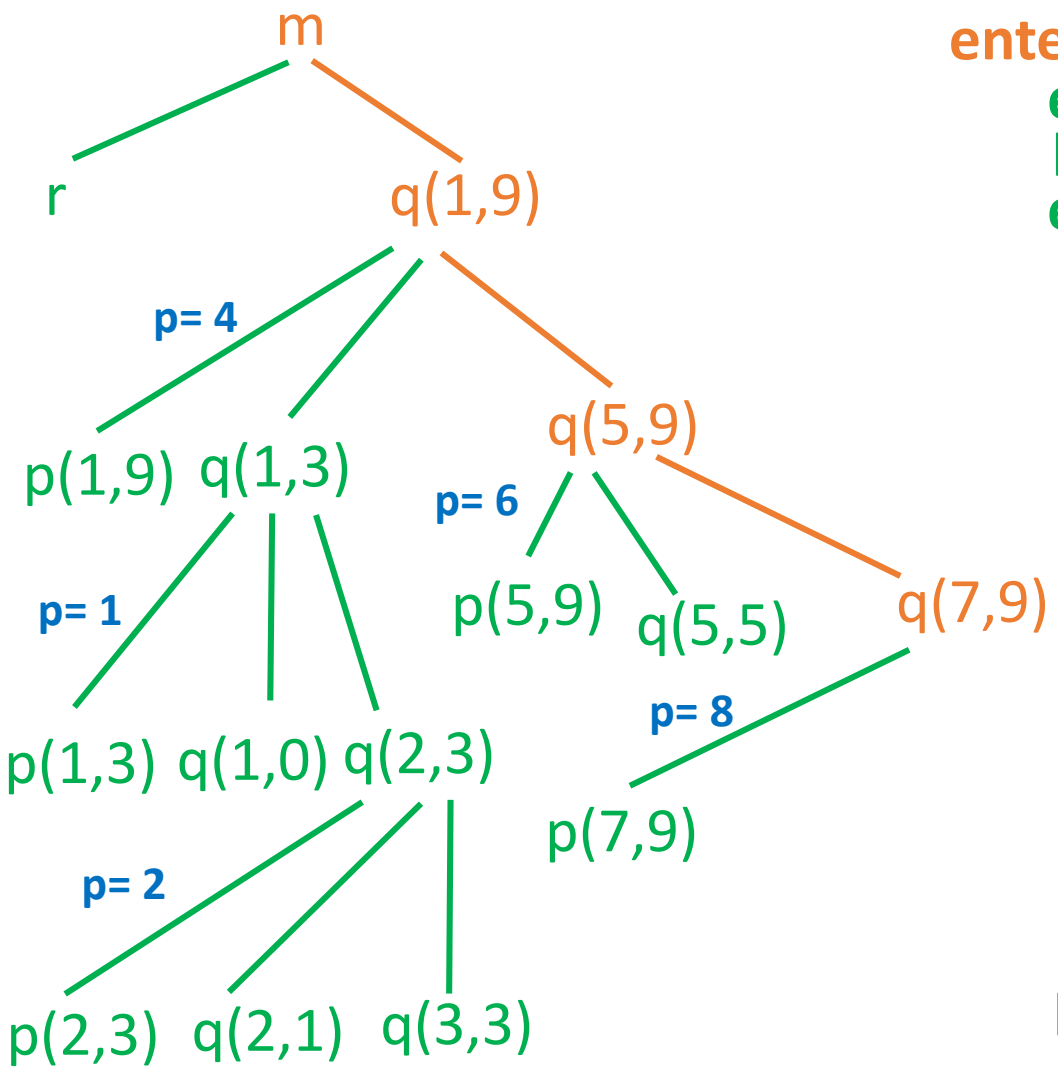
Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)

enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

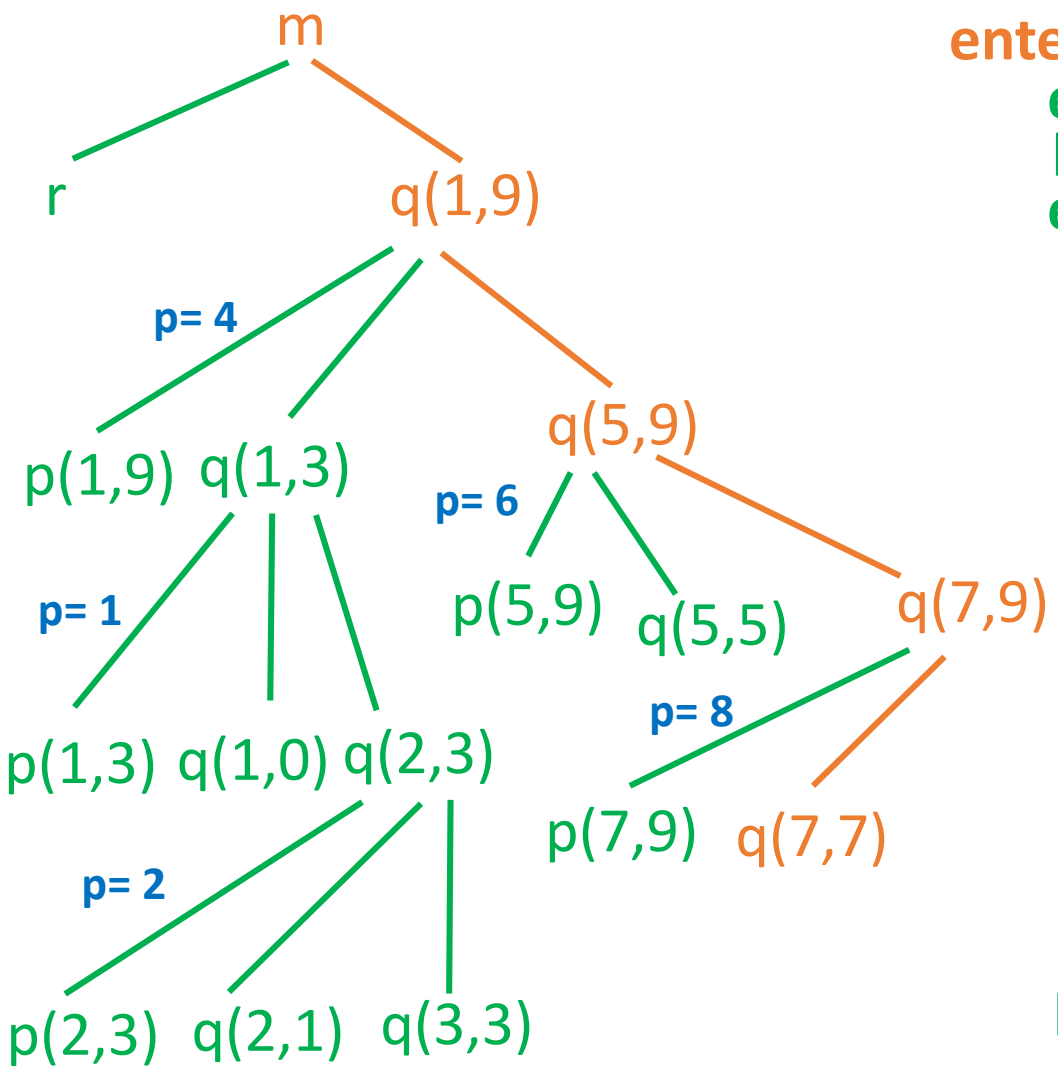


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)



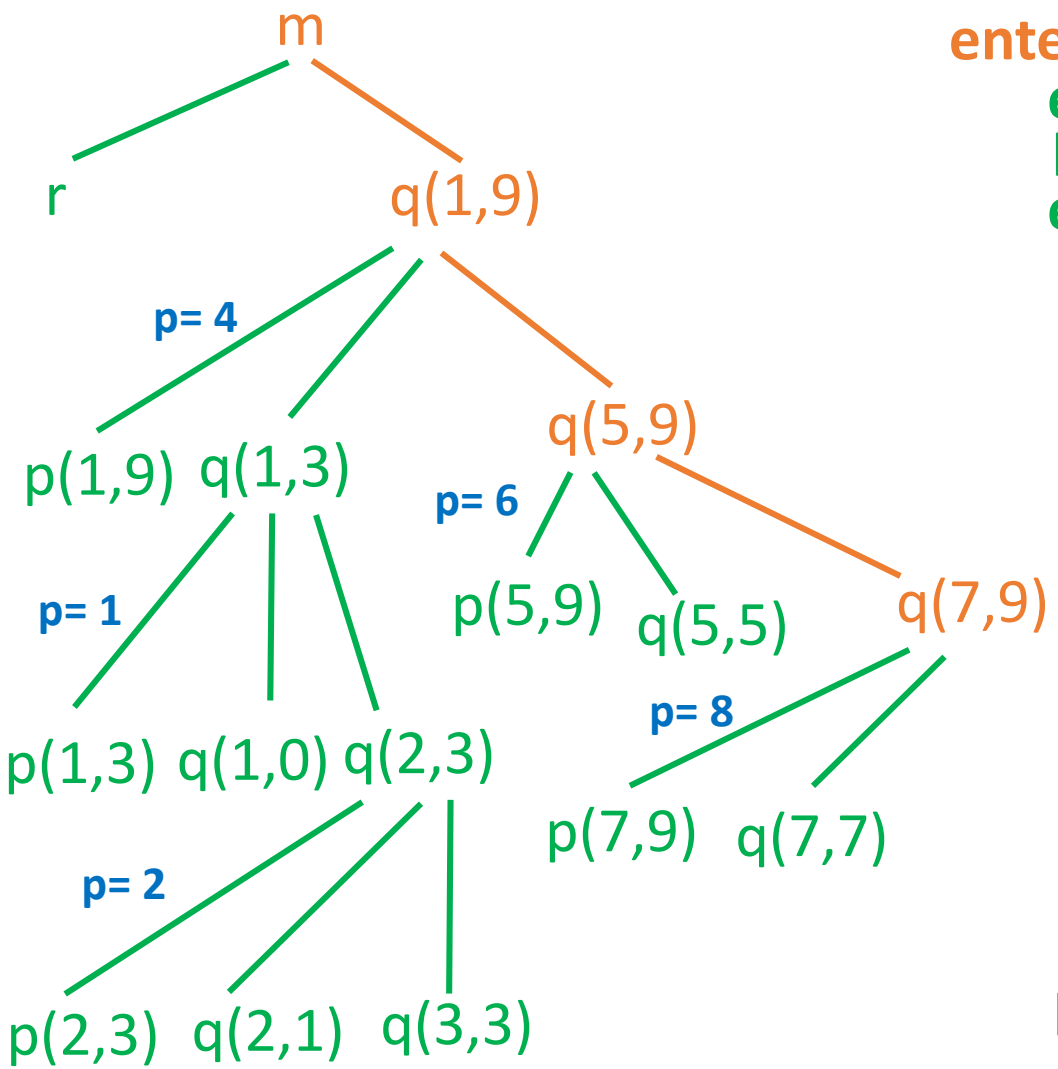
Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)

enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

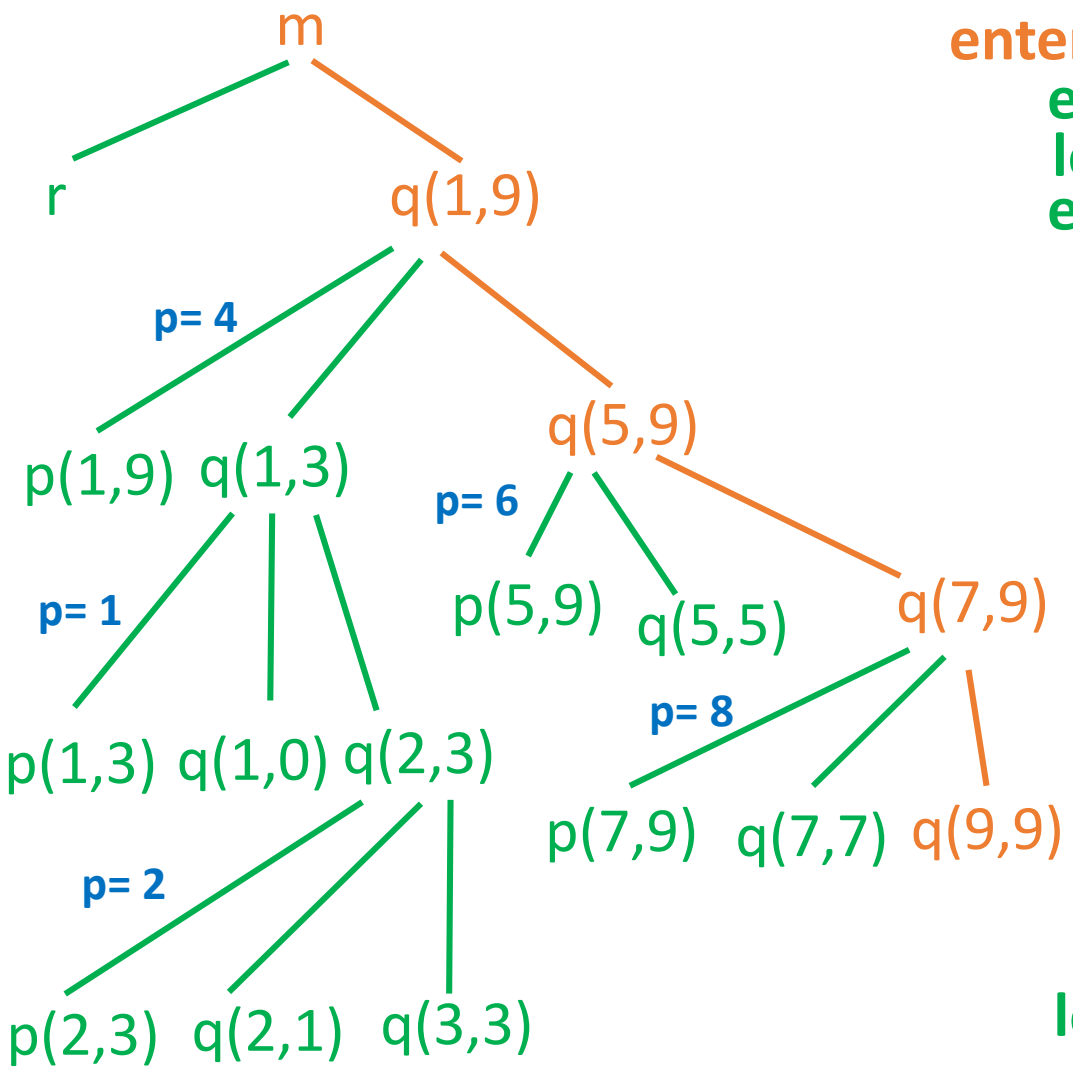


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)



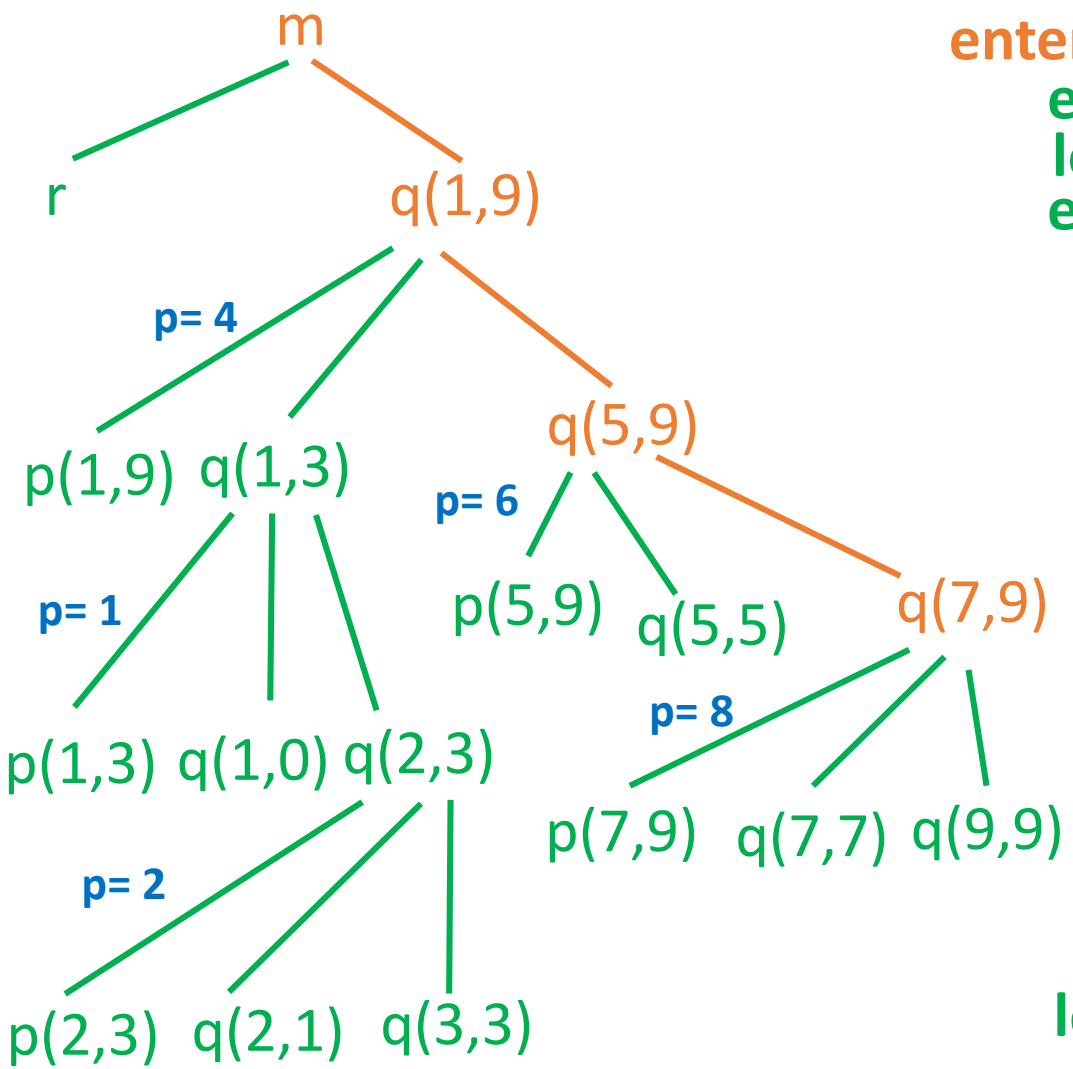
Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)

enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont...)

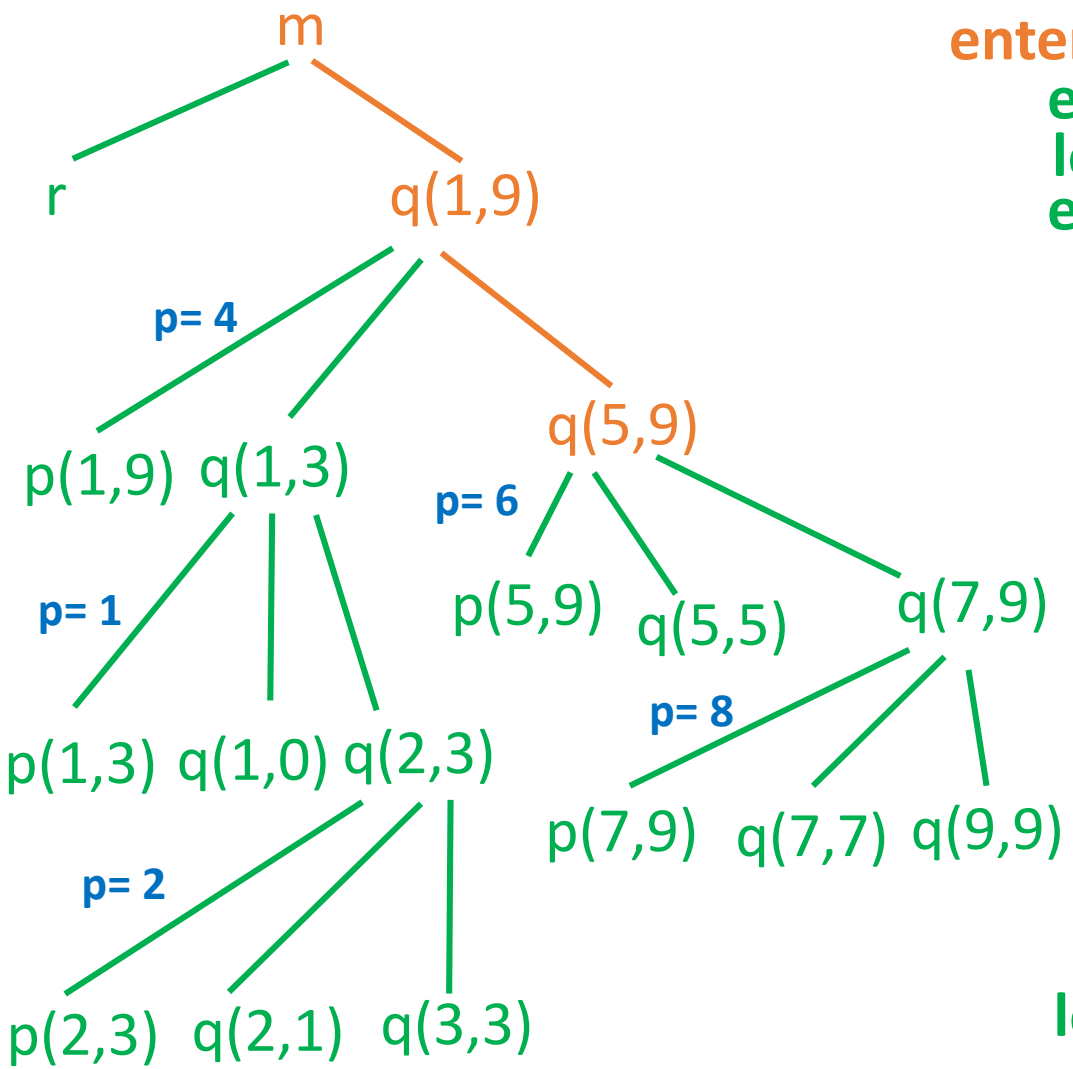


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9) p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont…)



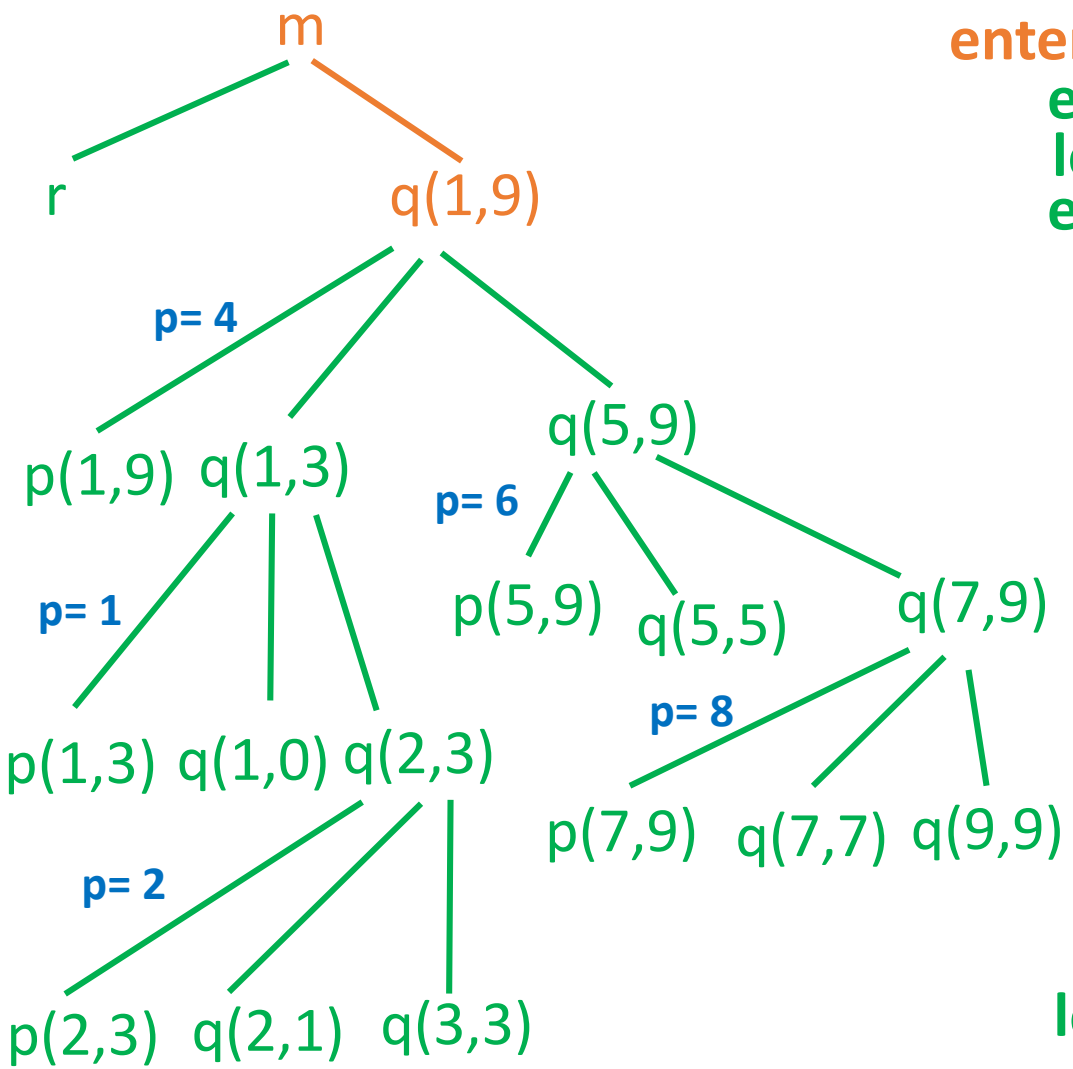
Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)

enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9)
p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont…)

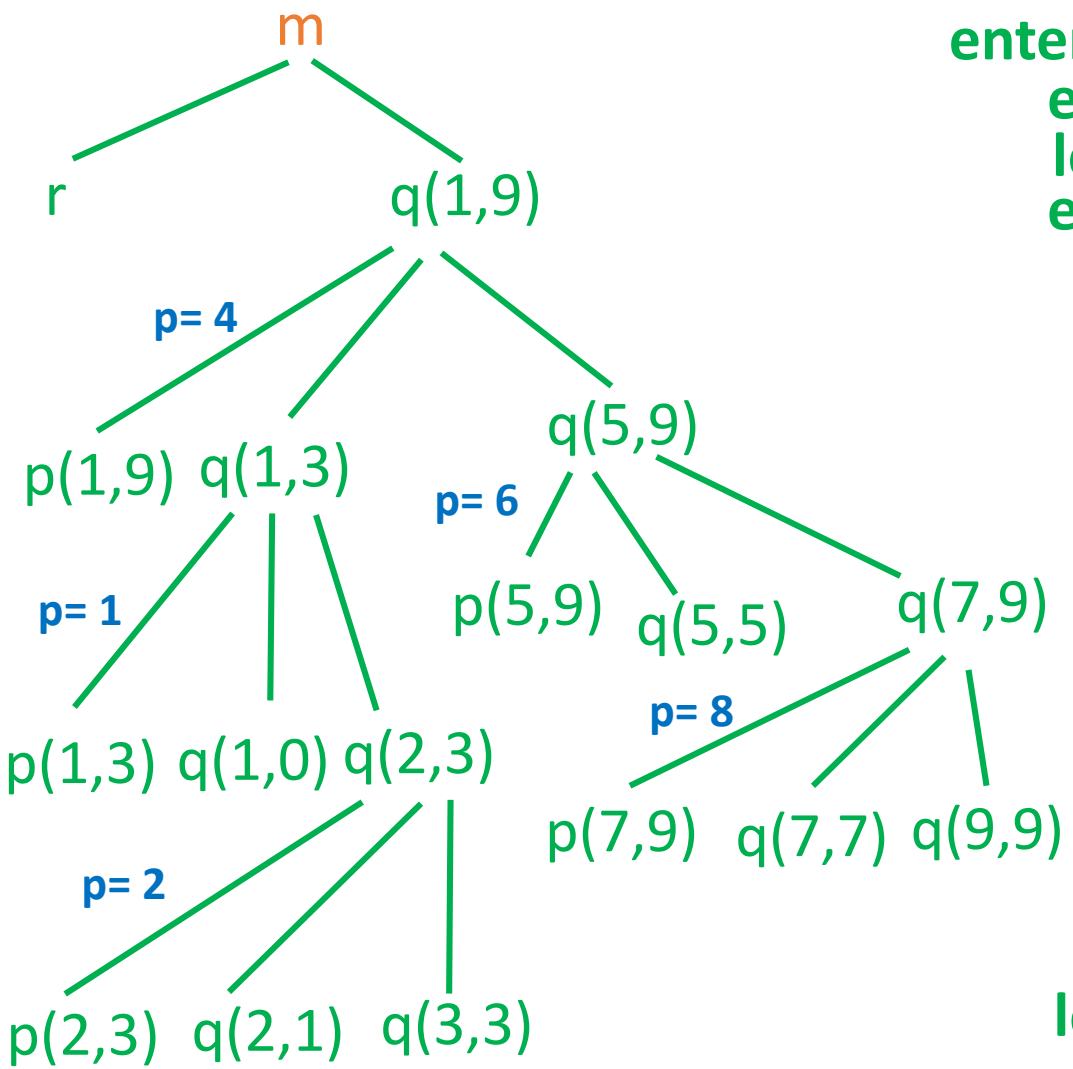


Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9)
p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont…)



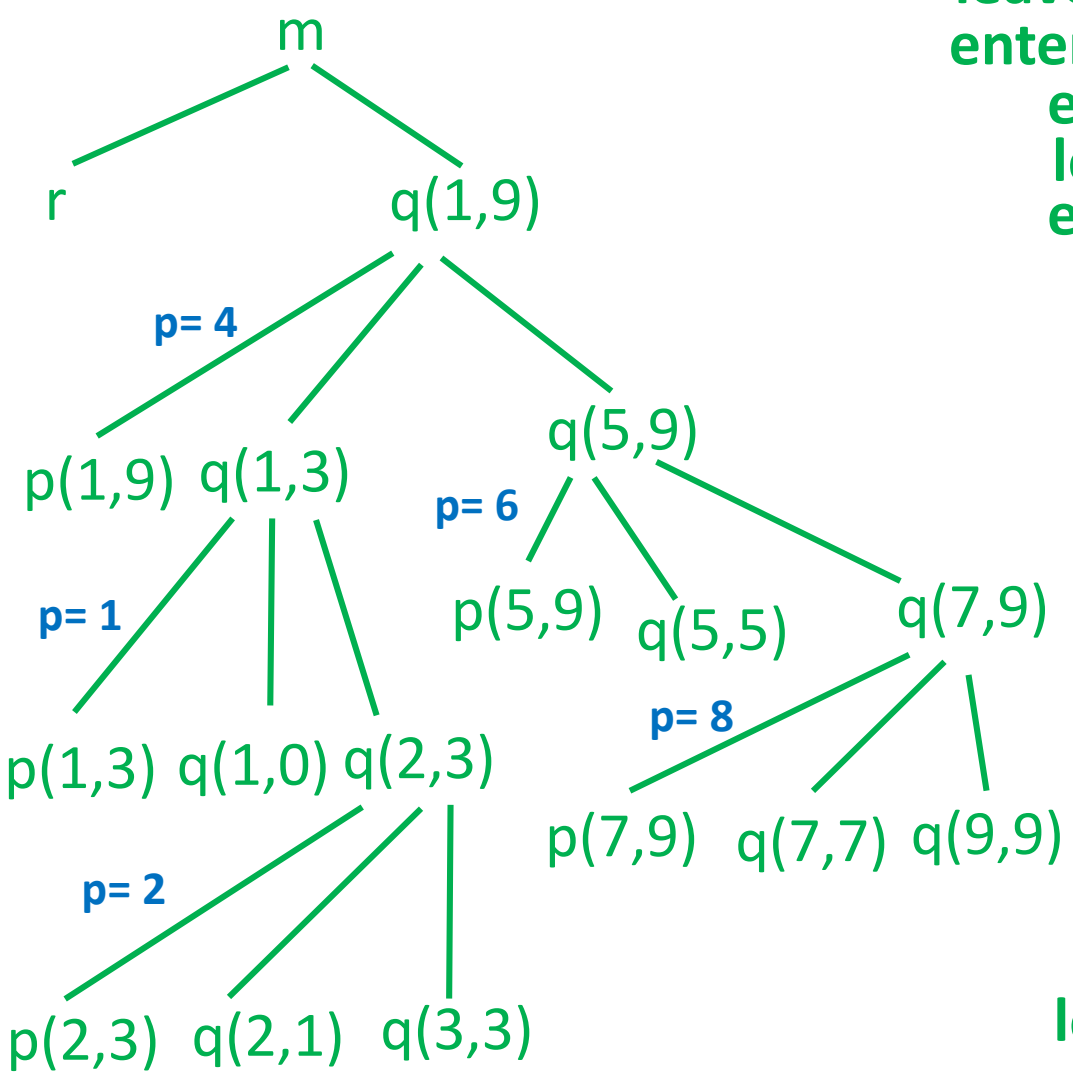
Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)

enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9)
p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Example 7.2(Cont…)



Activation Tree Representing Calls During an Execution of Quicksort

enter main ()
enter readArray ()
leave readArray ()
enter quicksort (1,9)
enter partition (1,9) → p= 4
leave partition (1,9)
enter quicksort (1,3)
enter partition (1,3) → p= 1
leave partition (1,3)
enter quicksort (1,0)
leave quicksort (1,0)
enter quicksort (2,3)
enter partition (2,3) → p= 2
leave partition (2,3)
enter quicksort (2,1)
leave quicksort (2,1)
enter quicksort (3,3)
leave quicksort (3,3)
leave quicksort (2,3)
leave quicksort (1,3)
enter quicksort (5,9)
enter partition (5,9) → p= 6
leave partition (5,9)
enter quicksort (5,5)
leave quicksort (5,5)
enter quicksort (7,9)
p= 8
enter partition (7,9)
leave partition (7,9)
enter quicksort (7,7)
leave quicksort (7,7)
enter quicksort (9,9)
leave quicksort (9,9)
leave quicksort (7,9)
leave quicksort (5,9)
leave quicksort (1,9)
leave main ()

Possible Activations for the Quicksort Program

Activation Trees (Cont…)

- **The use of a run-time stack is enabled by several useful relationships between the activation tree and the behavior of the program:**
  - The sequence of procedure calls corresponds to a preorder traversal of the activation tree
  - The sequence of returns corresponds to a postorder traversal of the activation tree

Activation Trees (Cont…)

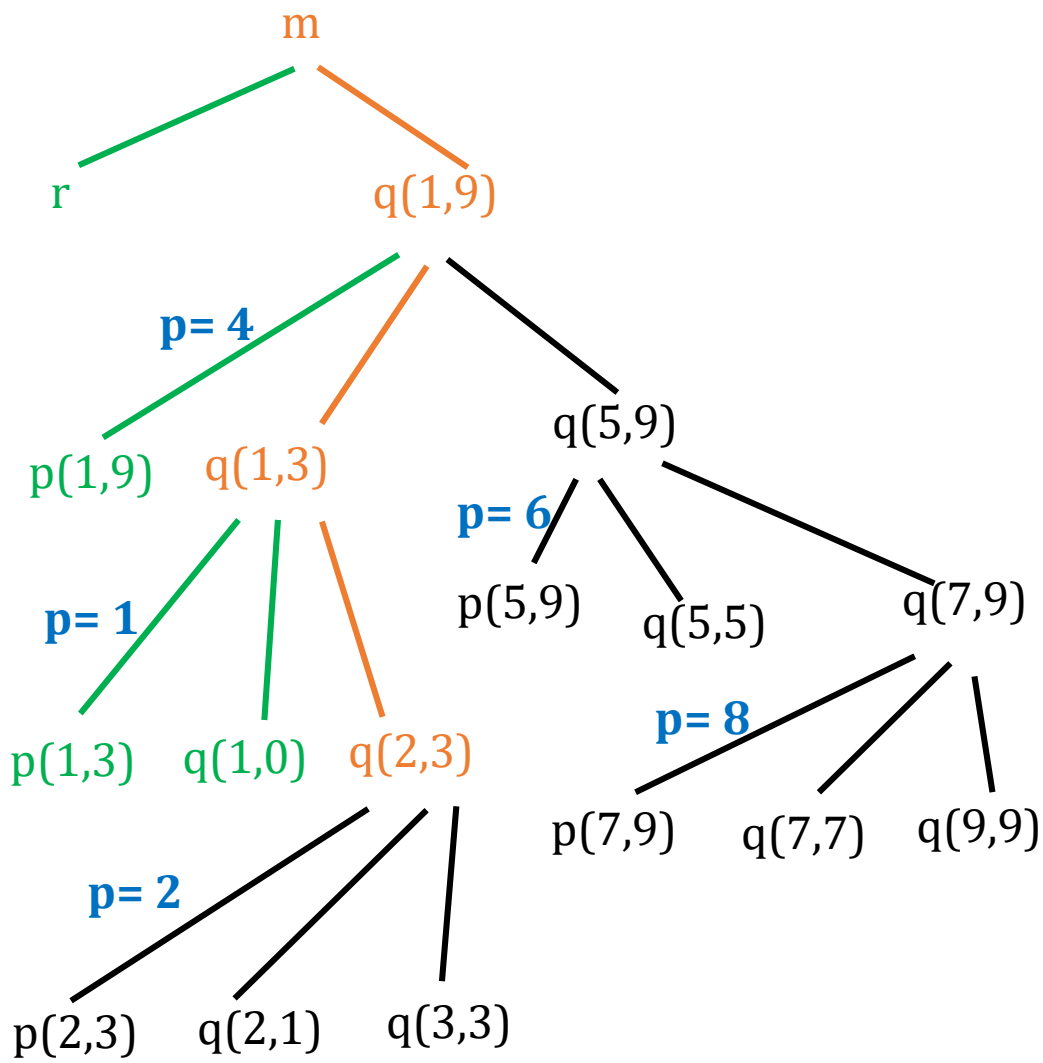
- **The use of a run-time stack is enabled by several useful relationships between the activation tree and the behavior of the program:**
- Suppose that control lies within a particular activation of some procedure corresponding to a node N of the activation tree
- Then the activations that are currently open (live) are those that correspond to node N and its ancestors
- The order in which these activations were called is the order in which they appear along the path to N starting at the root and they will return in the reverse of that order

Activation Records

- Procedure calls and returns are usually managed by a run-time stack called the control stack
- Each live activation has an activation record(sometimes called a frame) on the control stack
  - With the root of the activation tree at the bottom and
  - The entire sequence of activation records on the stack corresponding to the path in the activation tree to the activation where control currently resides
- The latter activation has its record at the top of the stack

Example 7.3

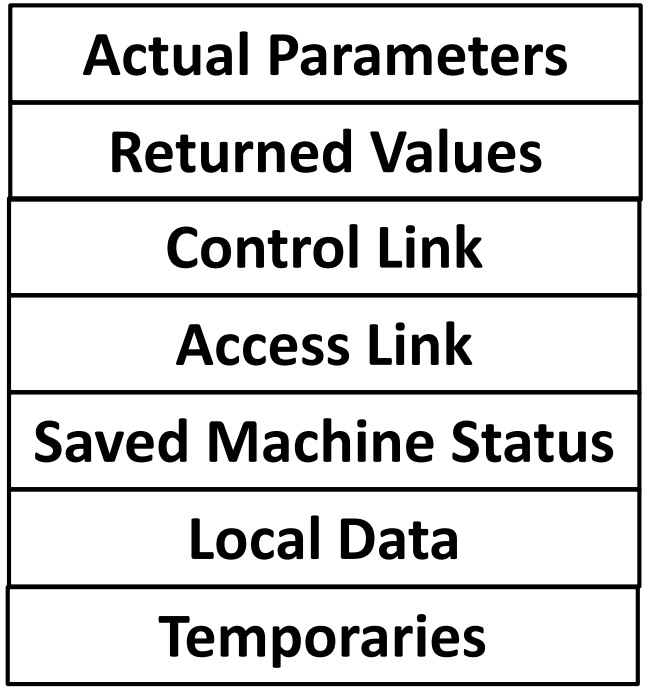
- If control is currently in the activation q(2,3) of the tree, then the activation record for q(2,3) is at the top of the control stack
- Just below is the activation record for q(1,3), the parent of q(2,3) in the tree
- Below that is the activation record for q(1,9), and
- At the bottom is the activation record for m, the main function and root of the activation tree



Activation Tree Representing Calls During an Execution of Quicksort

Contents of an Activation Record

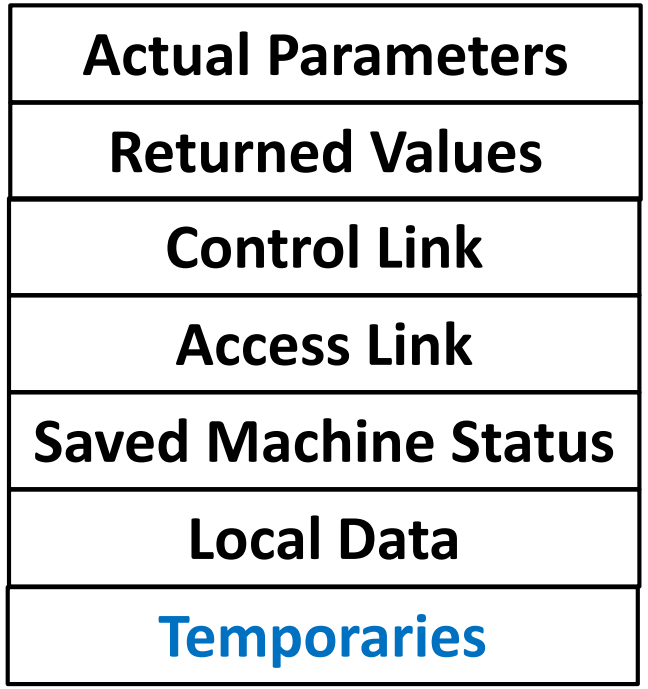
- The contents of activation records vary with the language being implemented
- Here is a list of the kinds of data that might appear in an activation record
- See figure for a summary and possible order for these elements



A General Activation Record

Contents of an Activation Record (Cont…)

- **Temporary values** such as those arising from the evaluation of expressions in cases where those temporaries cannot be held in registers
- For example, compiler generated temporaries like $\mathbf{t_1}$, $\mathbf{t_2}$ used to hold temporary values will be stored in **temporary** section



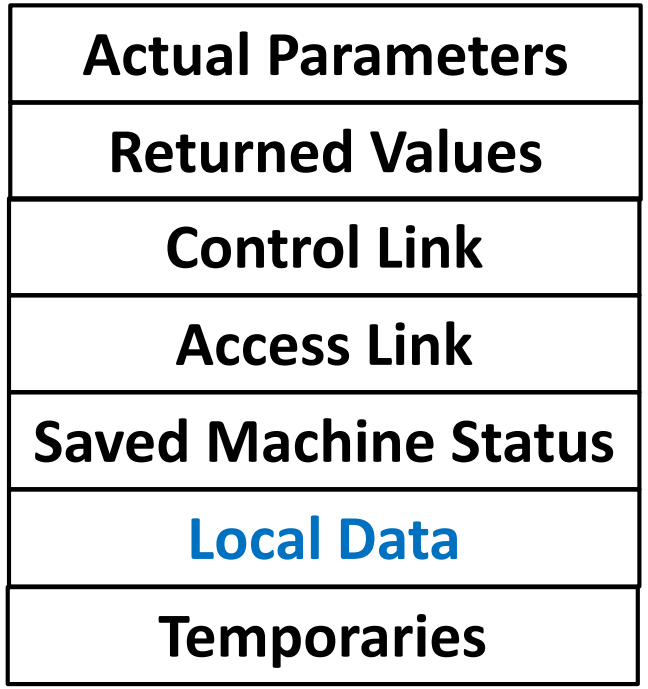
A General Activation Record

Contents of an Activation Record (Cont…)

- **Local data** belonging to the procedure whose activation record this is
- For example,

```
int sumfunc (int a, int b)
                                {
        int sum = a + b;
        return sum;
  }
```

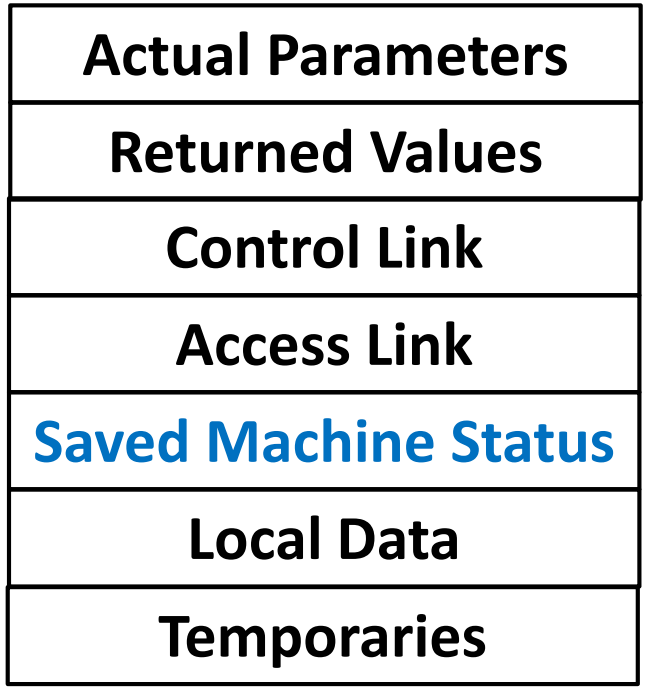
- **sum** is a local variable of the **sumfunc** function and it will be stored in **local data** section



A General Activation Record

Contents of an Activation Record (Cont…)

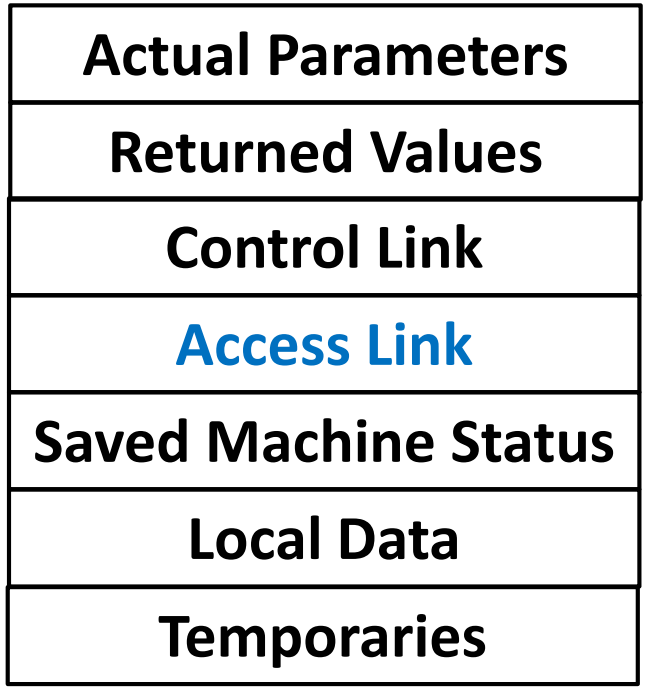
- A **saved machine status** with information about the state of the machine just before the call to the procedure
- This information typically includes the return address (For example, value of the program counter to which the called procedure must return) and
- The contents of registers that were used by the calling procedure and that must be restored when the return occurs



A General Activation Record

Contents of an Activation Record (Cont…)

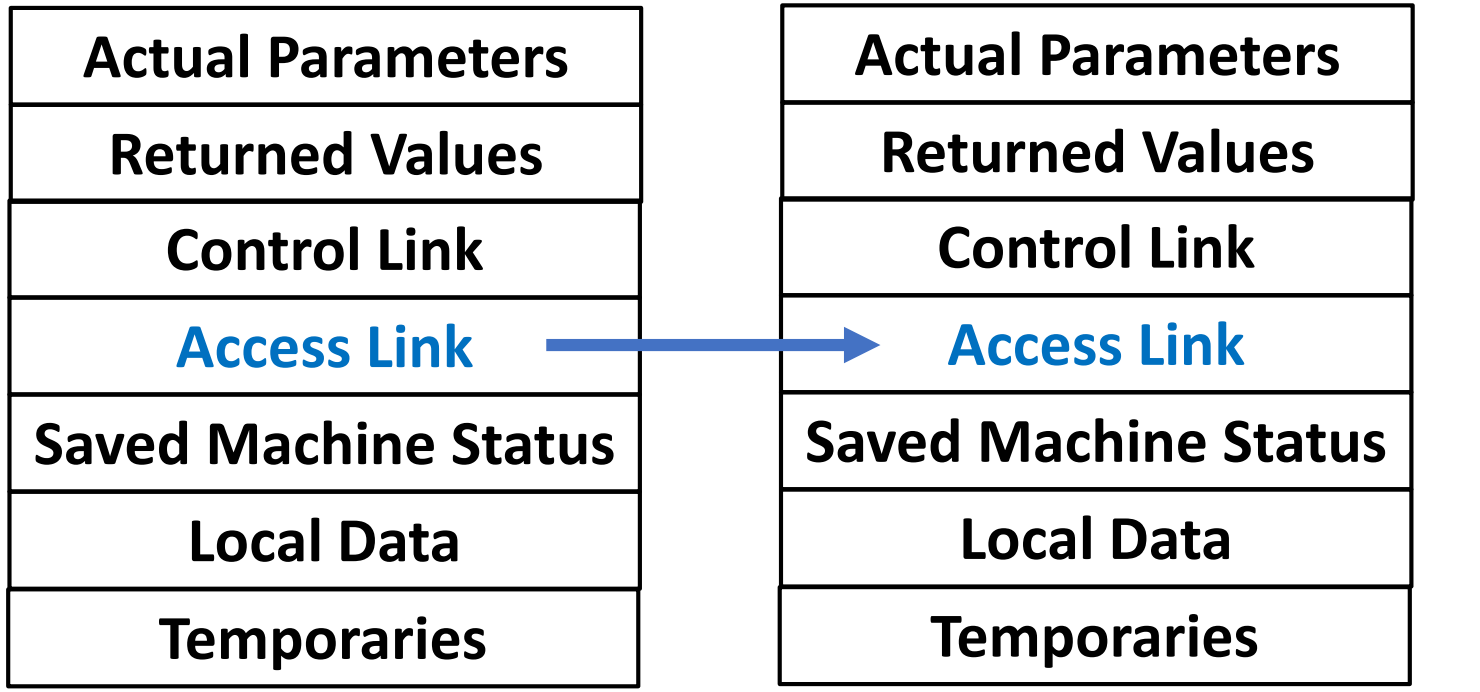
- An **access link** which is a pointer may be needed to locate data needed by the called procedure but found elsewhere e.g. in another activation record



A General Activation Record

Contents of an Activation Record (Cont…)

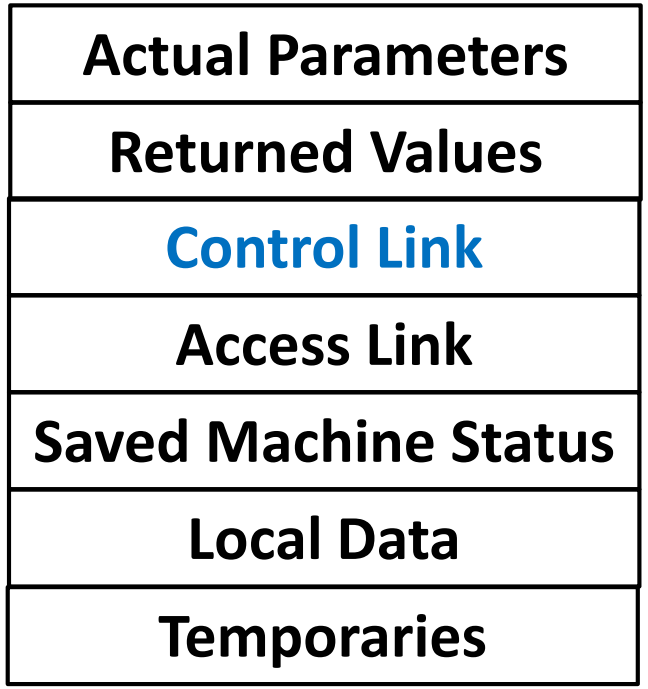
- For example, procedure **p()** needs data which is stored in the activation record of procedure **q()**
- Then the access link of **p()** will point to the access link of **q()**



Activation Record of p() **Activation Record of q()**

Contents of an Activation Record (Cont…)

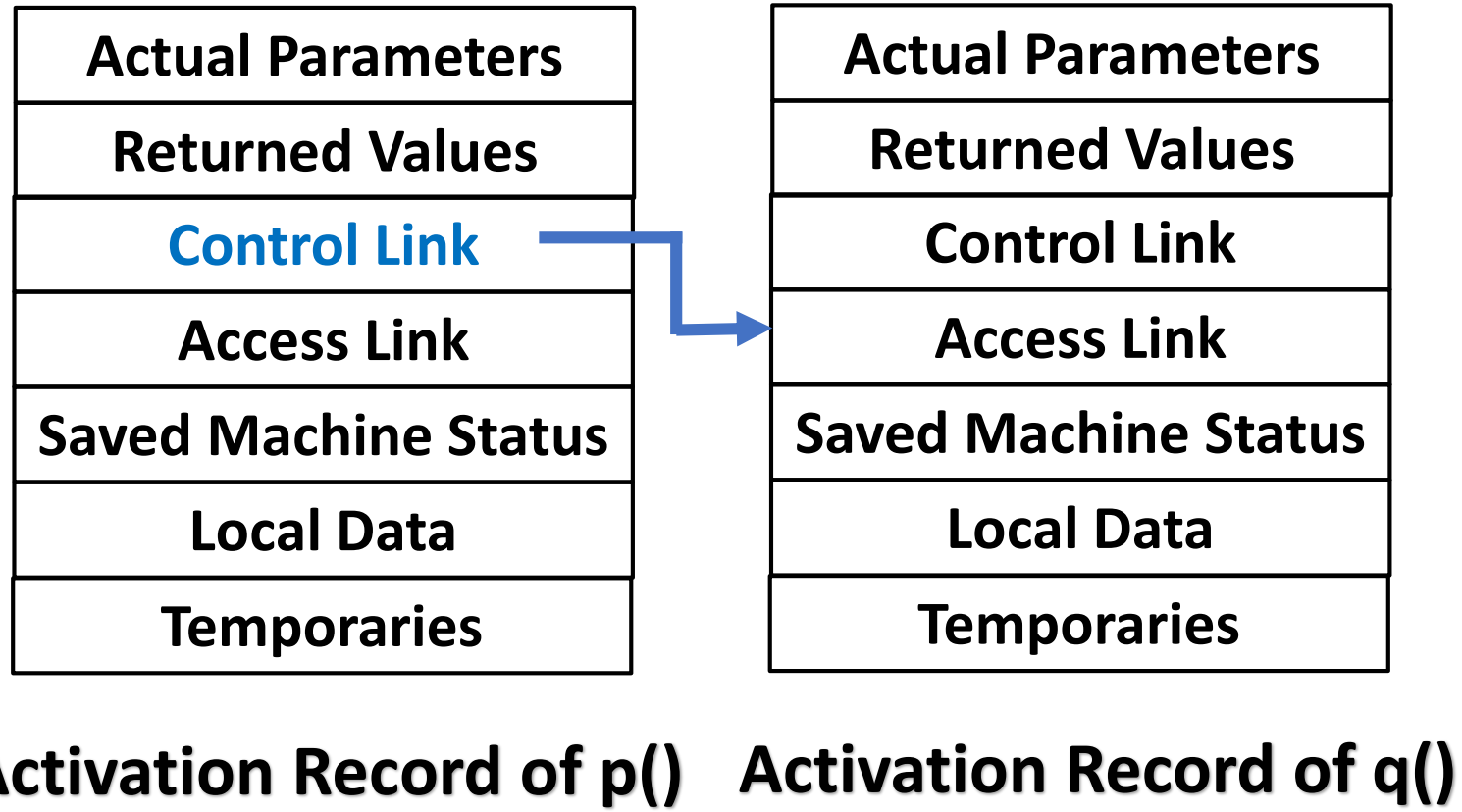
- A **control link** which is another pointer pointing to the activation record of the caller



A General Activation Record

Contents of an Activation Record (Cont…)

- For example, procedure **q()** calls procedure **p()**
- Then the control link of **p()** will point to the activation record of **q()**

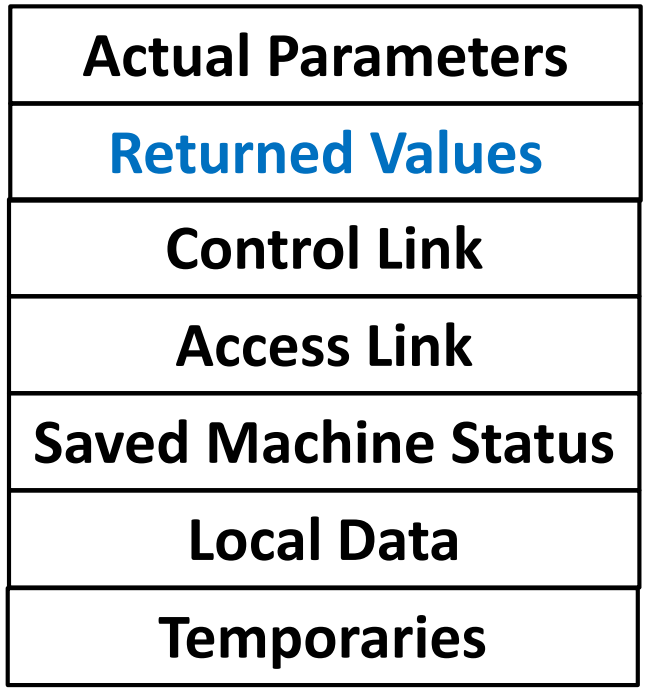


Contents of an Activation Record(Cont…)

- Space for the **return value** of the called function, if any as not all called procedures return a value
- For example,

```
int sumfunc (int a, int b)
 {
      int sum = a + b;
      return sum;
 }
```

- The **sumfunc** function returns value of **sum** and it will be stored in **return value** section but we may prefer to place that value in a register for efficiency



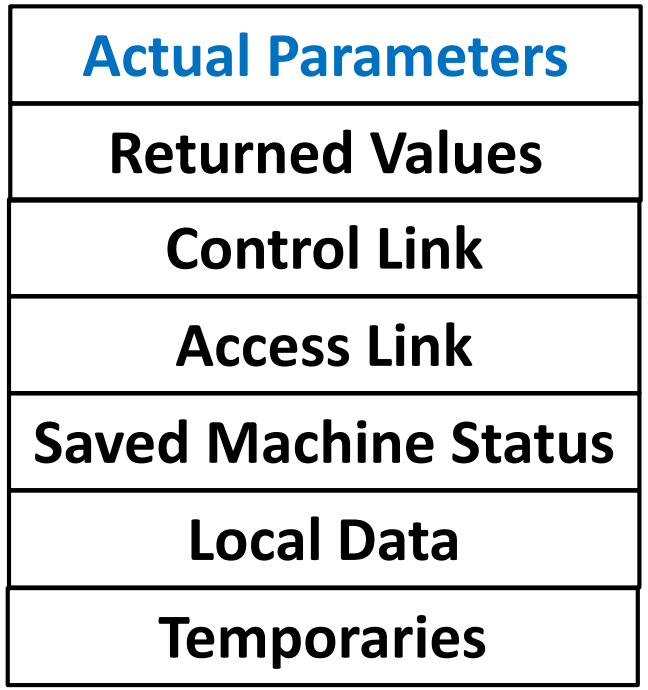
A General Activation Record

Contents of an Activation Record (Cont…)

- The **actual parameters** used by the calling procedure
- For example,

```
int sumfunc (int a, int b)
                                {
        int sum = a + b;
        return sum;
  }
```

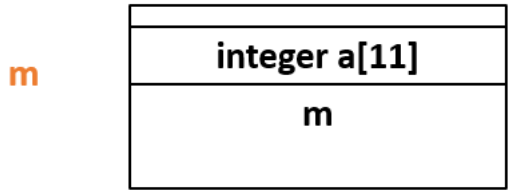
- **a** and **b** are the actual parameters of the **sumfunc** function and it will be stored in **actual parameters** section but we may prefer to place that value in a register for efficiency



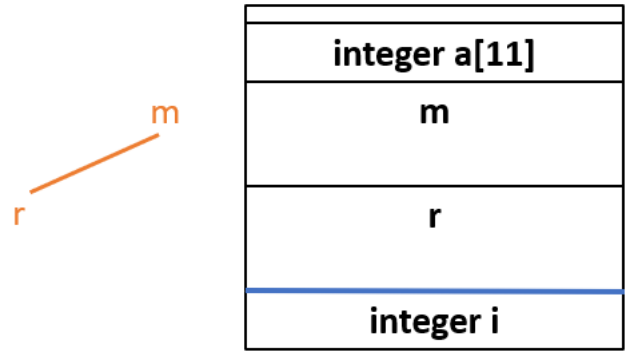
A General Activation Record

Example 7.4

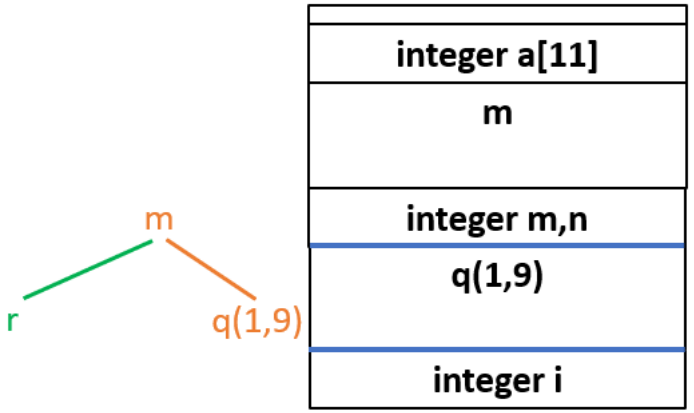
- Figure shows several snapshots of the run-time stack as control flows through the activation tree
- Green lines in the partial trees go to activations that have ended and orange lines indicates they are currently activated



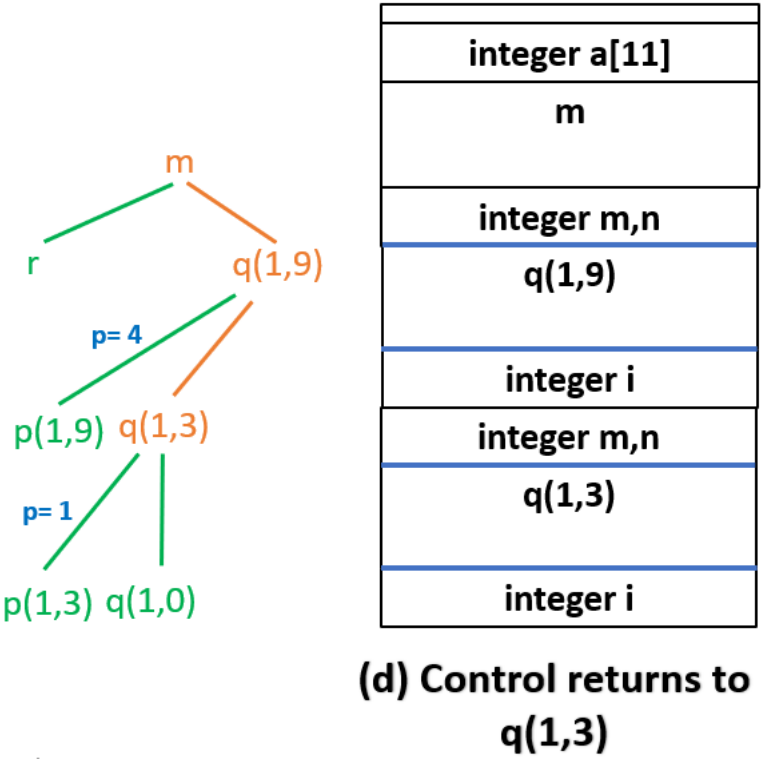
(a) Frame for main



(b) r is activated



(c) r has been popped and q(1,9) pushed



(d) Control returns to q(1,3)

Downward-growing stack of Activation Records

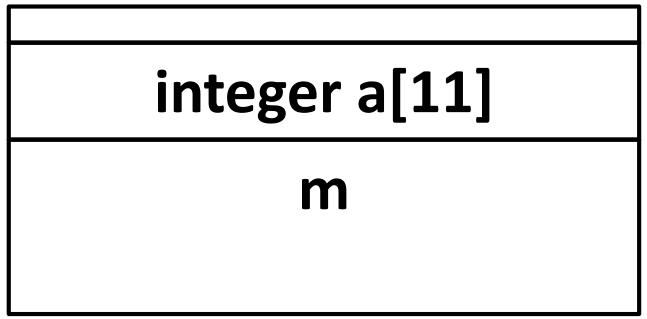
Example 7.4 (Cont...)

```
int a[11];
void readArray() { /* Reads 9 integers into a[1], ..., a[9]. */
    int i;
    ...
}
int partition(int m, int n) {
    /* Picks a separator value v, and partitions a[m..n] so that
       a[m..p-1] are less than v, a[p] = v, and a[p+1..n] are
       equal to or greater than v. Returns p. */
    ...
}
void quicksort(int m, int n) {
    int i;
    if (n > m) {
        i = partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
    }
}
main() {
    readArray();
    a[0] = -9999;
    a[10] = 9999;
    quicksort(1,9);
}
```

Sketch of a quicksort program

Since array a is global, space is allocated for it before execution begins with an activation of procedure main as shown in (a)

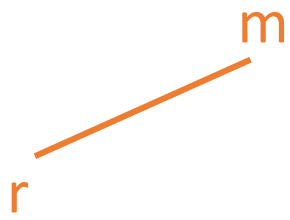
m

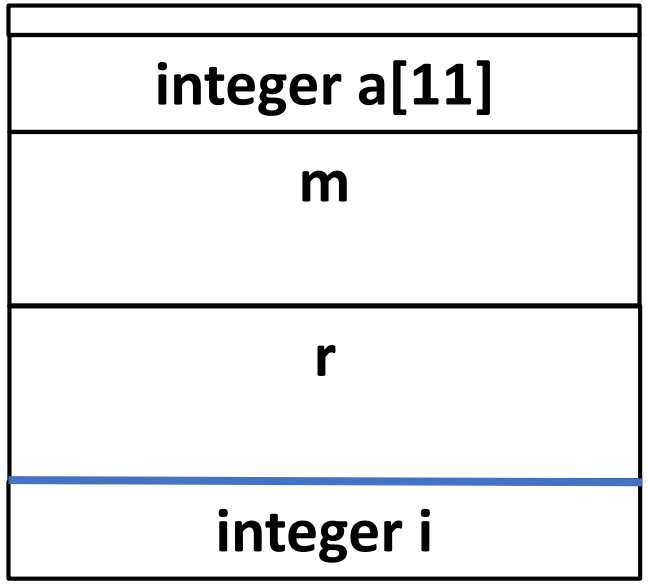


(a) Frame for main

Example 7.4 (Cont…)

- When control reaches the first call in the body of main procedure r is activated and its activation record is pushed onto the stack (b)
- The activation record for r contains space for local variable i
- The top of stack is at the bottom of diagrams

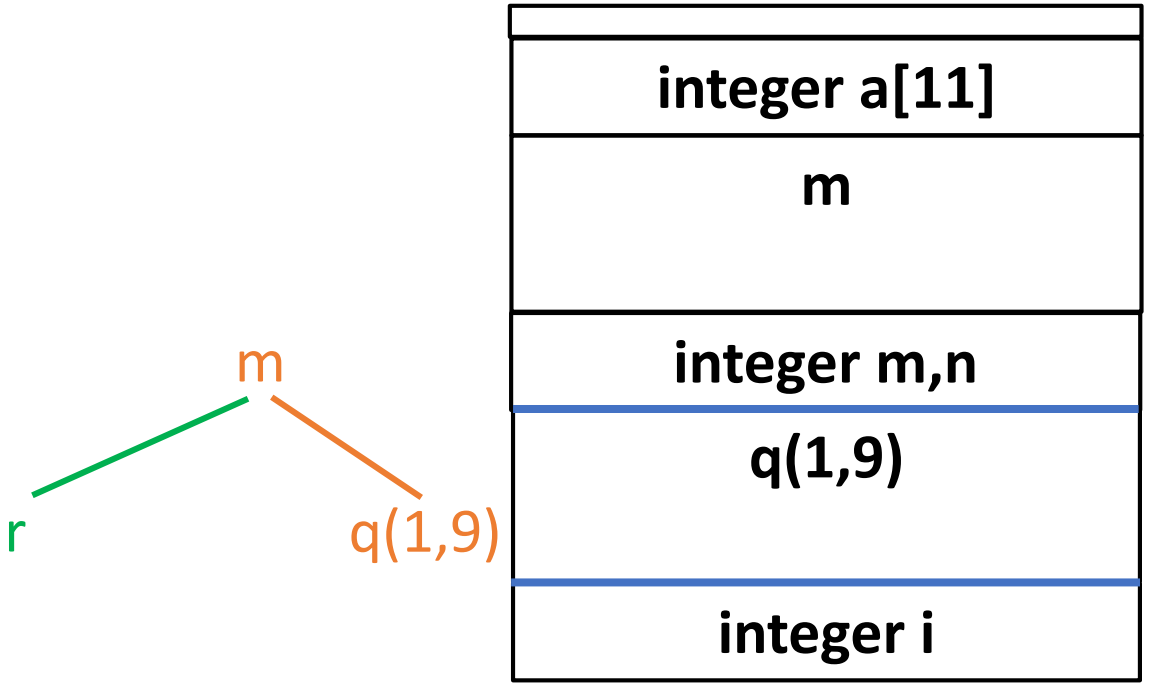




(b) r is activated

Example 7.4 (Cont…)

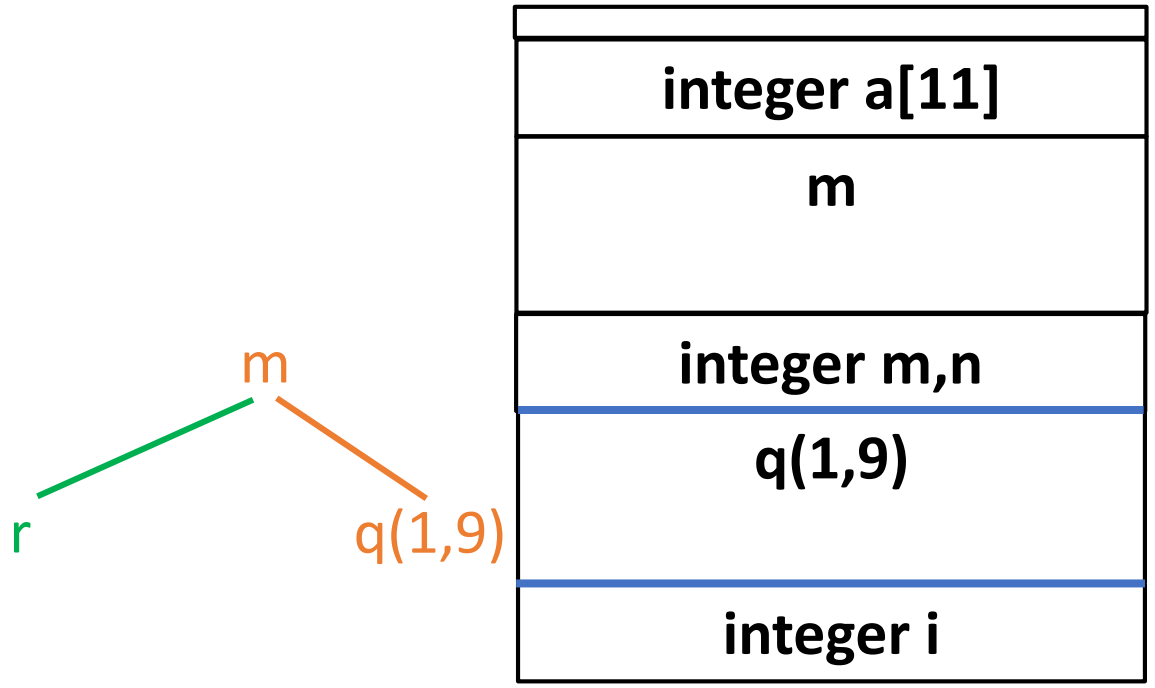
- When control returns from this activation its record is popped leaving just the record for main on the stack
- Control then reaches the call to q (quicksort) with actual parameters 1 and 9, and an activation record for this call is placed on the top of the stack as in (c)
- The activation record for q contains space for the parameters m and n and the local variable i following the general layout



(c) r has been popped and q(1,9) pushed

Example 7.4 (Cont…)

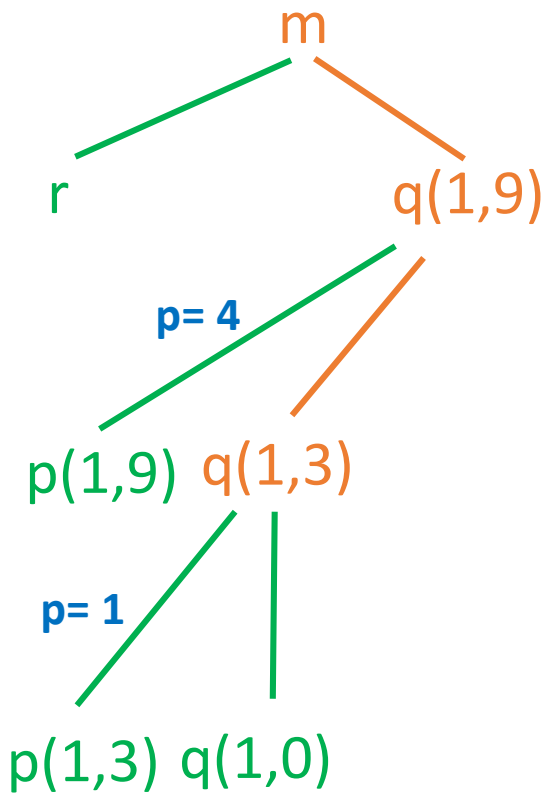
- Notice that space once used by the call of r is reused on the stack
- No trace of data local to r will be available to q(1,9)
- When q(1,9) will return then the stack again will hold only the activation record for main

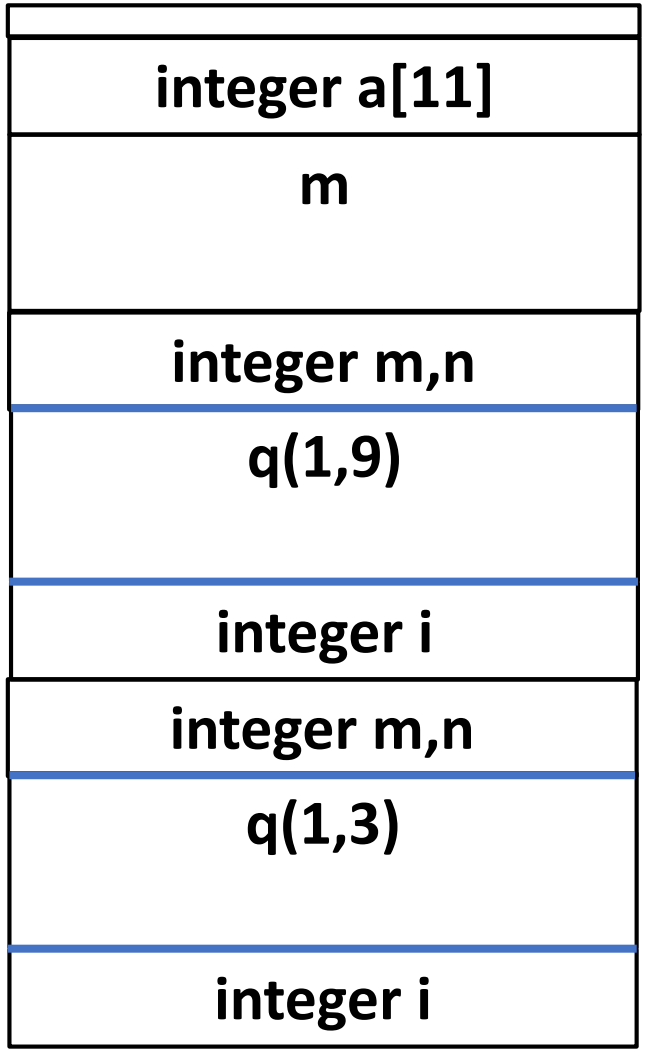


(c) r has been popped and q(1,9) pushed

Example 7.4 (Cont…)

- Several activations occur between the last two snapshots
- A recursive call to q(1,3) was made
- Activations p(1,3) and q(1,0) have begun and ended during the lifetime of q(1,3) leaving the activation record for q(1,3) on top of (d)
- Notice that when a procedure is recursive it is normal to have several of its activation records on the stack at the same time





(d) Control returns to q(1,3)

Calling Sequences

1. Values communicated between caller and callee are generally placed at the beginning of the callee's activation record.
2. Fixed-length items are generally placed in the middle which includes the control link, the access link, and the machine status fields.
3. Items whose size may not be known early enough are placed at the end of the activation record
4. Must locate the top-of-stack pointer to have it point to the end of the fixed-length fields in the activation record.

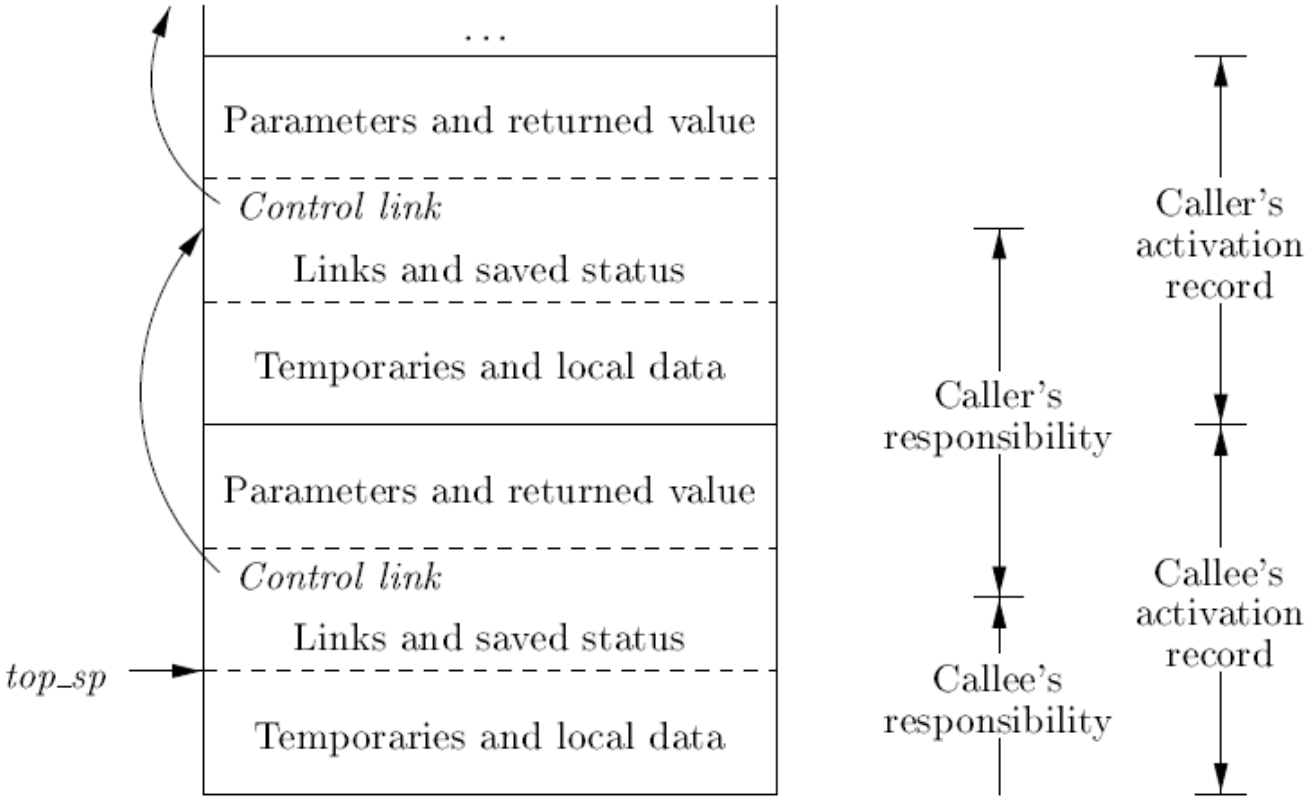


Figure 7.7: Division of tasks between caller and callee

Variable-Length Data on the Stack

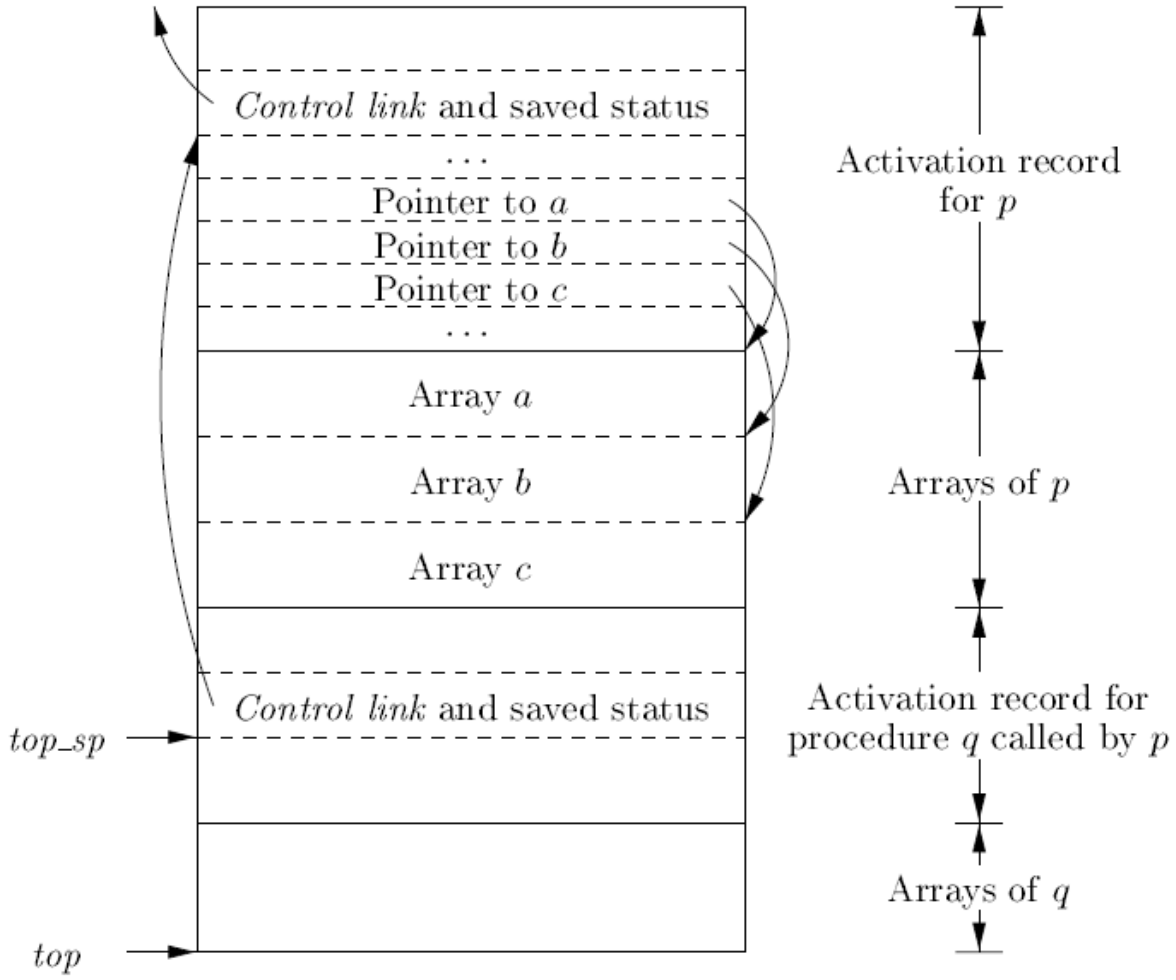


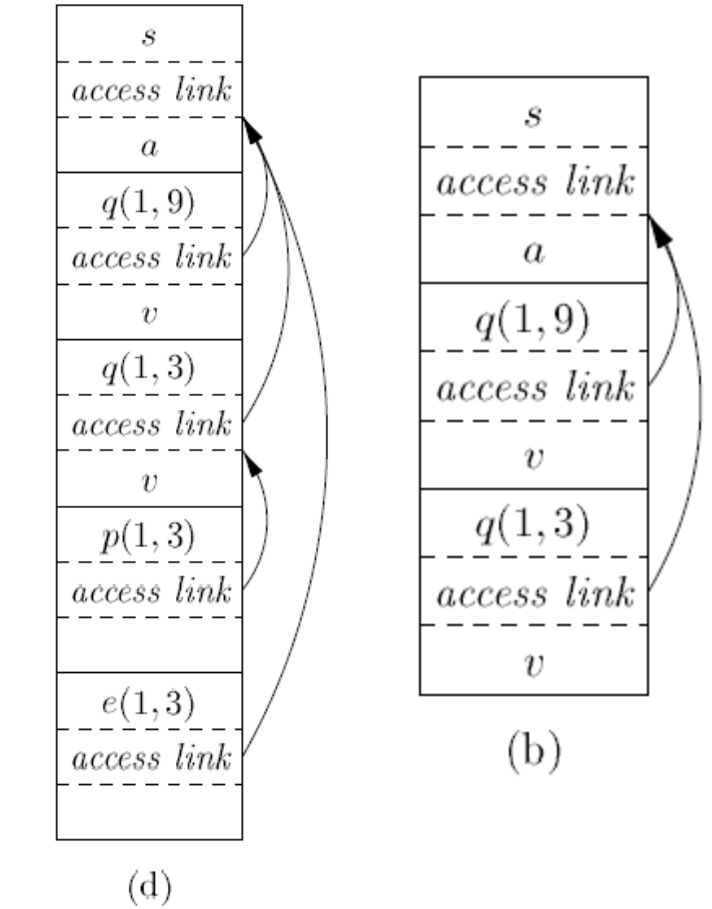
Figure 7.8: Access to dynamically allocated arrays

Access to Nonlocal Data on the Stack

- Data Access Without Nested Procedures
- Issues With Nested Procedures
- Nesting Depth
- Access Link
- Access Link with

Quicksort using Nested functions

```
1) fun sort(inputFile, outputFile) =
       let
2)         val a = array(11,0);
3)         fun readArray(inputFile) = ···
4)               ··· a ··· ;
5)         fun exchange(i,j) =
6)               ··· a ··· ;
7)         fun quicksort(m,n) =
               let
8)                 val v = ··· ;
9)                 fun partition(y,z) =
10)                     ··· a ··· v ··· exchange ···
               in
11)                ··· a ··· v ··· partition ··· quicksort
               end
       in
12)        ··· a ··· readArray ··· quicksort ···
       end;
```



calling these functions

Figure 7.10: A version of quicksort, in ML style, using nested functions

Quicksort using Nested functions

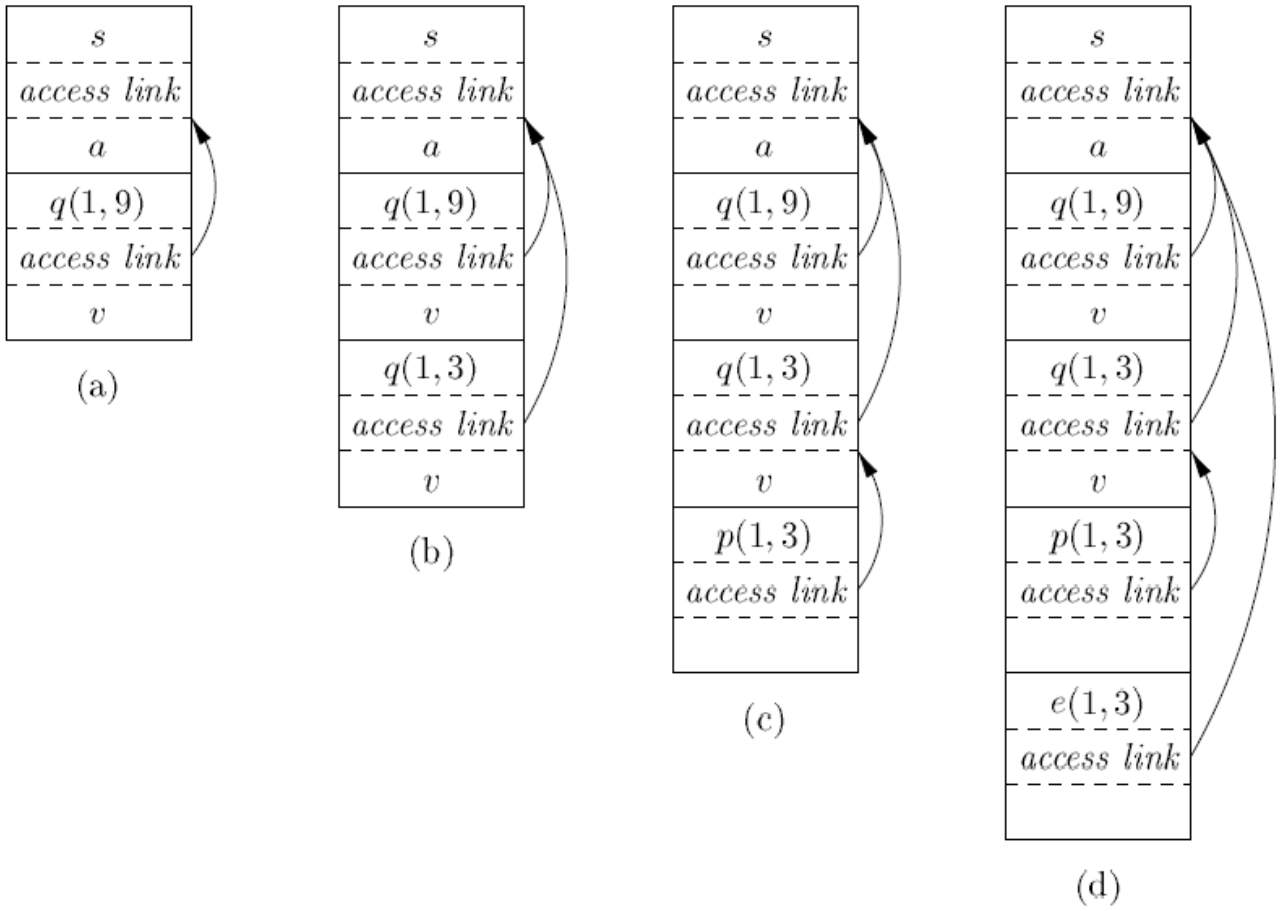
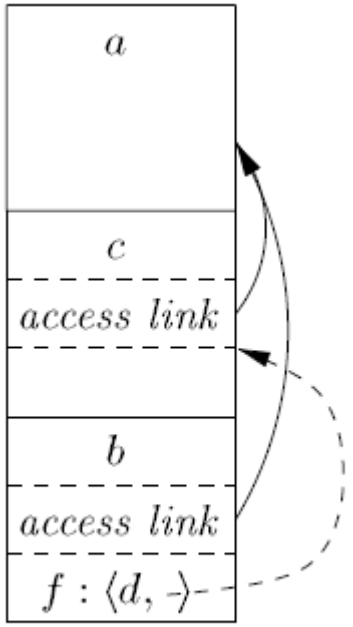


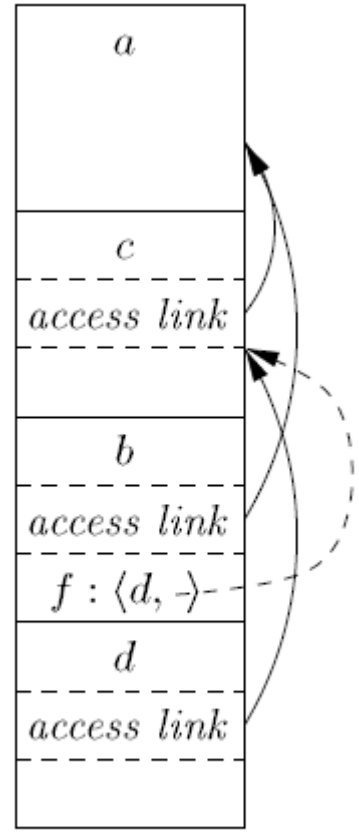
Figure 7.11: Access links for finding nonlocal data

Access Links for Procedure Parameters

```
fun a(x) =
    let
        fun b(f) =
            ··· f ··· ;
        fun c(y) =
            let
                fun d(z) = ···
            in
                ··· b(d) ···
            end
    in
        ··· c(1) ···
    end;
```



(a)



(b)

END