Extracting Performance Functions
Basic Operations

- The number of Basic Operations performed must be proportional to the run time
- Counting techniques depend on control structures
- The Worst Case assumption is most common
- Average Case can be done for some algorithms
Basic Operations: Examples

- Sorting  Key-to-Key Comparisons
- Searching  Key-to-Unknown Compares
- Matrix Multiply  Adds, or Multiplies
- Graph Operations  Processing a Vertex
- Polynomial Evaluation  Arithmetic Ops.
Counting Procedures

- **Straight-Line Code:**
  - Simply Count the operations you see

- **Assume basic operation is addition**

```
  a := b + c - d
  c := x + 5
  d := d * 3
  e := e + 1
```

Total Operations = 3
Counting IF Statements

- **Basic Operation is Addition**
- **Assume Worst Case**
- **Count Only One Side**

If \( a = b + c \) Then
\[
    c := d + e \\
    f := g + c + 1
\]
Else
\[
    c := e + f
\]
Endif

Total Operations = 4
Counting Loops

- Assume Worst-Case Number of Iterations
- Count Body, Multiply by Iteration Count
- Assume Basic Operation is Addition

For $i := 1$ to $12$ do
  $a := b + c$
  $d := d + 7$
End For

Total Operations $= 24$
Input Dependent Loops

- If the number of iterations depends on the size of the input, \( n \), then the count is a function of \( n \)

\[
\text{For } i := 1 \text{ to } n \text{ do}
\]
\[
\text{a := b+c}
\]
\[
d := d + 7
\]

End For

Total Operations = 2\( n \)
The following rules of thumb usually apply

- A single loop yields a linear function of \( n \)
- A doubly-nested loop yields a function of \( n^2 \)
- A triply-nested loop yields a function of \( n^3 \)

Be Careful when applying these rules

For \( i := 1 \) to \( n \) do
  For \( j := 1 \) to \( n \) do
    \( a := a + 1 \)
  End For
End For
Total Operations = \( n^2 \)

For \( i := 1 \) to \( n \) do
  For \( j := 1 \) to 3 do
    \( a := a + 1 \)
  End For
End For
Total Operations = \( 3n \)
It is necessary to take the peculiarities of an algorithm into account when counting operations.

\[
\begin{align*}
i &:= 1; \text{Cond} := \text{ExternalFunction}() ; \\
\text{While (}i<n\text{) And (Cond) do} & \\
\quad & a := a + 1; \\
\quad & \text{Cond} := \text{ExternalFunction}() ; \\
\quad & i := i + 1; \\
\text{End While;} & \\
\text{For } j := i \text{ to } n \text{ do} & \\
\quad & a := a + 1; \\
\text{End For;} & \\
\text{Total Operations} &= n
\end{align*}
\]
Recursive Functions

- Define $W(n)$ as the number of operations done for input of size $n$
- When encountering a recursive call, add $W(x)$ where $x$ is the size of the input for the recursive call
- More work must be done to obtain a usable solution
Recursion: An Example

- Basic Operation is Multiplication
- Size of input is Value of $x$

Function Fact($x$: integer):Integer
begin
  If $x < 1$ Then
    Fact = 1;
  Else
    Fact := Fact($x-1$)*$x$;
  End If
end

$W(n) = W(n-1) + 1$
The Equation $W(n) = W(n-1) + 1$ is called a Recurrence Relation. It must be solved to remove the reference to $W$ on the right hand side. Solution requires a boundary condition of the form $W(a) = k$ for constants $a$ and $k$. In the *Fact* example: $W(0) = 0$. 

**Boundary Conditions**
Extracting Performance Functions
Basic Operations

- The number of Basic Operations performed must be proportional to the run time
- Counting techniques depend on control structures
- The Worst Case assumption is most common
- Average Case can be done for some algorithms
Basic Operations: Examples

- Sorting Key-to-Key Comparisons
- Searching Key-to-Unknown Compares
- Matrix Multiply Adds, or Multiplies
- Graph Operations Processing a Vertex
- Polynomial Evaluation Arithmetic Ops.
Counting Procedures

- Straight-Line Code:
  - Simply Count the operations you see
- Assume basic operation is addition

\[
\begin{align*}
  a & := b + c - d \\
  c & := x + 5 \\
  d & := d \times 3 \\
  e & := e + 1
\end{align*}
\]

Total Operations = 3
Counting IF Statements

- Basic Operation is Addition
- Assume Worst Case
- Count Only One Side

If $a = b+c$ Then
  \[ c := d + e \]
  \[ f := g + c + 1 \]
Else
  \[ c := e + f \]
Endif

Total Operations = 4
Counting Loops

- Assume Worst-Case Number of Iterations
- Count Body, Multiply by Iteration Count
- Assume Basic Operation is Addition

For $i := 1$ to 12 do
  $a := b + c$
  $d := d + 7$
End For

Total Operations $= 24$
Input Dependent Loops

If the number of iterations depends on the size of the input, $n$, then the count is a function of $n$

For $i := 1$ to $n$ do
  a := b+c
  d := d + 7
End For

Total Operations = $2n$
The following rules of thumb usually apply
- A single loop yields a linear function of $n$
- A doubly-nested loop yields a function of $n^2$
- A triply-nested loop yields a function of $n^3$

Be Careful when applying these rules

For $i := 1$ to $n$ do
  For $j := 1$ to $n$ do
    $a := a + 1$
  End For
End For
Total Operations = $n^2$

For $i := 1$ to $n$ do
  For $j := 1$ to 3 do
    $a := a + 1$
  End For
End For
Total Operations = $3n$
Algorithm Peculiarities

- It is necessary to take the peculiarities of an algorithm into account when counting operations.

```plaintext
i := 1; Cond := ExternalFunction( );
While (i<n) And (Cond) do
    a := a + 1;
    Cond := ExternalFunction( );
    i := i + 1;
End While;
For j := i to n do
    a := a + 1;
End For;
Total Operations = n
```
Recursive Functions

- Define $W(n)$ as the number of operations done for input of size $n$
- When encountering a recursive call, add $W(x)$ where $x$ is the size of the input for the recursive call
- More work must be done to obtain a usable solution
Recursion: An Example

- Basic Operation is Multiplication
- Size of input is Value of $x$

Function Fact($x$: integer):Integer
begin
  If $x < 1$ Then
    Fact = 1;
  Else
    Fact := Fact($x-1$)*$x$;
  End If
end

$W(n) = W(n-1) + 1$
Boundary Conditions

- The Equation $W(n) = W(n-1) + 1$ is called a Recurrence Relation.
- It must be solved to remove the reference to $W$ on the right hand side.
- Solution requires a boundary condition of the form $W(a) = k$ for constants $a$ and $k$.
- In the *Fact* example: $W(0) = 0$. 