Taller de Análisis de Algoritmos

Prof. Luis Garreta

Ingeniería de Sistemas y Computación
Pontificia Universidad Javeriana – Cali

5 de agosto de 2017

1. Ordene las siguientes funciones por tasa de crecimiento. Indique cuales funciones crecen a la misma taza.

   $N,\ \sqrt{N},\ N^{1.5},\ N^2,\ N \log N,$
   $N \log \log N,\ N \log^2 N,\ N \log(N^2),$
   $2/N,\ 2^N,\ 2^{N/2},\ 37,\ N^2 \log N,\ N^3.$

2. Calcule el tiempo de ejecución de los siguientes fragmentos de código. Para cada punto diga si la complejidad del algoritmos es $O(n), \Theta(n), \Omega(n)$ o es una combinación de los tres.

(a)
```
sum = 0;
for (i=1; i<=n; i++)
  sum += n;
```

(b)
```
int sum( int n ) {
  int partialSum;
  partialSum = 0;
  for( int i = 1; i <= n; ++i )
    partialSum += i * i * i;
  return partialSum;
}
```

(c)
```
for( i = 0; i < n; ++i )
  for( j = 0; j < n; ++j )
    ++k;
```

(d)
```
for( i = 0; i < n; ++i)
  a[ i ] = 0;

for( i = 0; i < n; ++i)
  for( j = 0; j < n; ++j )
    a[ i ] += a[ j ] + i + j;
```

(e)
```
sum = 0;
for (i=1; i<=n; i++)       // First for loop
  for (j=1; j<=i; j++)  // is a double loop
    sum++;
for (k=0; k<n; k++)        // Second for loop
  A[k] = k;
```

(f)
```
sum1 = 0;
for (i=1; i<=n; i++)  // First double loop
  for (j=1; j<=n; j++) //  do n times
    sum1++;

sum2 = 0;
for (i=1; i<=n; i++) // Second double loop
  for (j=1; j<=i; j++) //  do i times
    sum2++;
```

(g)
```
sum1 = 0;
for (k=1; k<=n; k*=2)
  for (j=1; j<=n; j++)
    sum1++;

sum2 = 0;
for (k=1; k<=n; k*=2)
  for (j=1; j<=k; j++)
    sum2++;
```

(h)
```
// Assume that the array A is ordered
int binary (int A[], int n, int K) {
  int l = -1;
  int r = n; //l and r are beyond array bounds
  while (l+1 != r) { // Stop when l and r meet
    int i = (l+r)/2; // Check middle of
                     // remaining subarray
    if (K == A[i]) return i; // Found it
    if (K < A[i]) r = i;     // In left half
    else (K > A[i]) l = i;     // In right half
  }
  return n; //   Search value not in A
}
```

(i)
```
long factorial( int n ) {
  if( n <= 1 )
    return 1;
  else
    return n * factorial( n - 1 );
}
```