Calculating complexity — Examples

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 2

Calculating complexity

- Iterative programs
- Recursive programs

Example 1

Find the maximum element in a list

- Input size is length of the list
- Single loop scans all elements
- Always takes n steps
- Overall time is $O(n)$

```
def maxElement(L):
  maxval = L[0]
  for i in range(len(L)):
    if L[i] > maxval:
      maxval = L[i]
return(maxval)
```

Example 2

Check whether a list contains duplicates

- Input size is length of the list
- Nested loop scans all pairs of elements
- A duplicate may be found in the very first iteration
- Worst case — no duplicates, both loops run fully
- Time is $(n-1)+(n-2)+\ldots+1 = n(n-1)/2$
- Overall time is $O(n^2)$

```
def noDuplicates(L):
  for i in range(len(L)):
    for j in range(i+1,len(L)):
      if L[i] == L[j]:
        return(False)
  return(True)
```

Example 3

Matrix multiplication

- Matrix is represented as list of lists
  - $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$
  - `[[1,2,3],[4,5,6]]`
- Input matrices have size $m \times n$, $n \times p$
- Output matrix is $m \times p$
- Three nested loops
- Overall time is $O(mnp)$ — $O(n^3)$ if both are $n \times n$

```
def matrixMultiply(A,B):
  (m,n,p) = (len(A),len(B),len(B[0]))

  C = [[ 0 for i in range(p) ]
           for j in range(m) ]

  for i in range(m):
    for j in range(p):
      for k in range(n):
        C[i][j] = C[i][j] + A[i][k]*B[k][j]

  return(C)
```

Example 4

Number of bits in binary representation of n

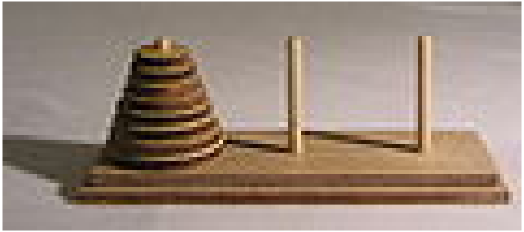
- $\log n$ steps for n to reach 1
- For number theoretic problems, input size is number of digits
- This algorithm is linear in input size

```
def numberOfBits(n):

  count = 1

  while n > 1:
    count = count + 1
    n = n // 2

  return(count)
```

Example 5

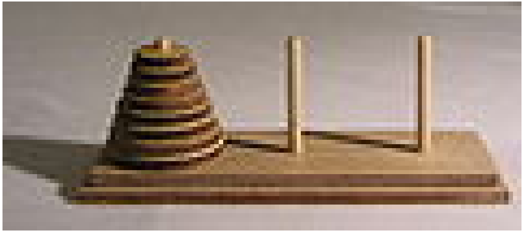
Towers of Hanoi

- Three pegs A,B,C
- Move n disks from A to B, use C as transit peg
- Never put a larger disk on a smaller one



Example 5

Towers of Hanoi



- Three pegs A,B,C
- Move n disks from A to B, use C as transit peg
- Never put a larger disk on a smaller one

Recursive solution

- Move $n-1$ disks from A to C, use B as transit peg
- Move larges disk from A to B
- Move $n-1$ disks from C to B, use A as transit peg

Example 5

Recurrence

- $M(n)$ — number of moves to transfer n disks
- $M(1) = 1$
- $M(n) = M(n-1) + 1 + M(n-1) = 2M(n-1) + 1$

Example 5

Recurrence

- $M(n)$ — number of moves to transfer n disks
- $M(1) = 1$
- $M(n) = M(n-1) + 1 + M(n-1) = 2M(n-1) + 1$

Unwind and solve

$$M(n) = 2M(n-1) + 1$$

Example 5

Recurrence

- $M(n)$ — number of moves to transfer n disks
- $M(1) = 1$
- $M(n) = M(n-1) + 1 + M(n-1) = 2M(n-1) + 1$

Unwind and solve

$$\begin{aligned} M(n) &= 2M(n-1) + 1 \\ &= 2(2M(n-2)+1) + 1 = 2^2 M(n-2) + (2+1) \end{aligned}$$

Example 5

Recurrence

- $M(n)$ — number of moves to transfer n disks
- $M(1) = 1$
- $M(n) = M(n-1) + 1 + M(n-1) = 2M(n-1) + 1$

Unwind and solve

$$\begin{aligned} M(n) &= 2M(n-1) + 1 \\ &= 2(2M(n-2) + 1) + 1 = 2^2 M(n-2) + (2+1) \\ &= 2^2(2M(n-3) + 1) + (2+1) = 2^3 M(n-3) + (4+2+1) \end{aligned}$$

Example 5

Recurrence

- $M(n)$ — number of moves to transfer n disks
- $M(1) = 1$
- $M(n) = M(n-1) + 1 + M(n-1) = 2M(n-1) + 1$

Unwind and solve

$$
\begin{aligned}
M(n) &= 2M(n-1)+1 \\
&= 2(2M(n-2)+1)+1 = 2^2M(n-2)+(2+1) \\
&= 2^2(2M(n-3)+1)+(2+1) = 2^3M(n-3)+(4+2+1) \\
&\quad \cdots \\
&= 2^kM(n-k)+(2^k-1))
\end{aligned}
$$

Example 5

Recurrence

- $M(n)$ — number of moves to transfer n disks
- $M(1) = 1$
- $M(n) = M(n-1) + 1 + M(n-1) = 2M(n-1) + 1$

Unwind and solve

$$
\begin{aligned}
M(n) &= 2M(n-1) + 1 \\
&= 2(2M(n-2) + 1) + 1 = 2^2M(n-2) + (2+1) \\
&= 2^2(2M(n-3) + 1) + (2+1) = 2^3M(n-3) + (4+2+1) \\
&\quad \cdots \\
&= 2^kM(n-k) + (2^k - 1)) \\
&\quad \cdots \\
&= 2^{n-1}M(1) + (2^{n-1} - 1)
\end{aligned}
$$

Example 5

Recurrence

- $M(n)$ — number of moves to transfer n disks
- $M(1) = 1$
- $M(n) = M(n-1) + 1 + M(n-1) = 2M(n-1) + 1$

Unwind and solve

$$
\begin{aligned}
M(n) &= 2M(n-1) + 1 \\
&= 2(2M(n-2) + 1) + 1 = 2^2 M(n-2) + (2+1) \\
&= 2^2(2M(n-3) + 1) + (2+1) = 2^3 M(n-3) + (4+2+1) \\
&\dots \\
&= 2^k M(n-k) + (2^k - 1)) \\
&\dots \\
&= 2^{n-1} M(1) + (2^{n-1} - 1) \\
&= 2^{n-1} + 2^{n-1} - 1 = 2^n - 1
\end{aligned}
$$

Summary

- Iterative programs
    - Focus on loops

Summary

- Iterative programs
  - Focus on loops
- Recursive programs
  - Write and solve a recurrence

Summary

- Iterative programs
  - Focus on loops
- Recursive programs
  - Write and solve a recurrence
- Need to be clear about accounting for "basic" operations