Running Times of Algorithms

CSE/IT 122 ~ Algorithms & Data Structures

Bubble Sort

- Algorithm

```
for i:= 1 to (n-1) // length of Array is n, A[1..n]
    for j:= n to (i+1)
    if A[j-1] > A[j]
        // swap A[j-1] < A[j]
        tmp := A[j-1]
        A[j-1] := A[j]
        A[j] = tmp
```

- Example
  - A = ⟨10,7,9,3,4⟩

Bubble Sort

- ➔ Example
  - A = ⟨10,7,9,3,4⟩
    - Test 5 times total when i = 1
    - Test 4 times when i = 2
    - Test 3 times when i = 3
    - Test 2 times when i = 2

Bubble Sort Analysis

➔ The first loop goes from 1 to (n-1) so that executes (n-1) times, plus 1 time for final check (so ((n-1)-1+1)+1 = n times)

Bubble Sort	cost	times
`for i := 1 to (n - 1)`	c_1	$n - 1 + 1 = n$
`  for j:= n to (i + 1)`	c_2	$\sum_{i=1}^{n-1}(n - i + 1) = n(n+1)/2 - 1$
`    if A[j-1] > A[j]`	c_3	$\sum_{i=1}^{n-1}(n - i) = n(n+1)/2 - 1$
`      tmp := A[j-1]`	c_4	?
`      A[j-1] := A[j]`	c_5	?
`      A[j] := tmp`	c_6	?

Bubble Sort Analysis

- Why the second for loop is

$$\sum_{i=1}^{n-1}(n-i+1) = n(n+1)/2 - 1$$

- The loop is going to execute n-(i+1)+1+1 times. If you just subtract n-(i+1) you have a fencepost error (off by one error). So you add one to correct for the count. The other one comes from the for loop test. And n-(i+1)+1+1 = n-i+1
- The limits of summation come from the first for loop
- Expand the summation and you can show

$$\sum_{i=1}^{n-1}(n-i+1) = (n-1+1) + (n-2+1) + \cdots + (n-(n-1)+1) = n + (n-1) + \cdots + 2$$

- So subtract 1 from the sum formula for the fact you are starting at 1 in the formula

Bubble Sort Analysis

- ➔ Best Case
  - Already sorted, swap never occurs
- ➔ Worst Case
  - Swap happens every time
- ➔ Average Case
  - Like worst case, involves n^2

Recursion

- ➔ A recurrence relation for the sequence $\{a_n\}$ is a formula that expresses a_n in terms of one or more of the previous terms of the sequence.
- ➔ You also need the initial conditions of the recurrence relation
- ➔ Example:
  - Assume $a_n = a_{n-1} - a_{n-2}$ for $n = 2,3,4,...$ and suppose $a_0 = 3$ and $a_1 = 5$. What are a_2 and a_3?
    - Can find by substitution
    - $a_2 = a_1 - a_0 = 5 - 3 = 2$
    - $a_3 = a_2 - a_1 = 2 - 5 = -3$

Rabbits

- A young pair of rabbits, one of each sex, is placed on an island. A pair of rabbits does not breed until they are two months old. After two months each pair of rabbits produces another pair each month. Find a recurrence relation for the number of pairs of rabbits on the island after n and assuming no rabbits ever die.

Rabbits

Month	Reproducing Pairs	Young Pairs	Total Pairs
1	0	1	1
2	0	1	1
3	1	1	2
4	1	2	3
5	2	3	5
6	3	5	8

- The sequence satisfies $f_n = f_{n-1} + f_{n-2}$ with $f_1 = f_2 = 1$
- This is, of course, the Fibonacci sequence

Solving Recurrences

- The goal is to take a recurrence relation and find an explicit formula
  - Example:
    - $a_k = a_{k-1}+2$
    - $a_0 = 1$
  - Then find an explicate formula using something like substitution or iteration
    - $a_0 = 1$
    - $a_1 = a_0 + 2 = 1+2 = 3$
    - $a_2 = a_1 + 2 = (1+2)+2 = 5$
    - $a_3 = a_2 + 2 = ((1+2)+2)+2 = 7$
    - … //generalize to n
    - $a_n = a_{n-1} + 2 = 1 + (n*2)$

Solving Recurrences

- This is a guess. To show it is correct you use … wait for it … MATHEMATICAL INDUCTION!!!
  - Base case: $a_0 = 2*0 + 1 = 1$ //true
  - Assume $a_k = 2k + 1$
  - Show $a_{k+1} = 2(k + 1) + 1 = 2k + 3$
  - Proof
    - $A_{k+1} = a_k + 2 = 2k + 1 + 2 = 2k + 3$

Solving Recurrences: Examples

➔ Find an explicit formula for $a_k = k*a_{k-1}$

Solving Recurrences: Examples

- Find an explicit formula for $a_k = k * a_{k-1}$
- Answer:
  - $A_n = n*(n-1)*(n-2)*3*2*1 = n!$