Test Review (and Homework)

<https://drive.google.com/open?id=0Bwxfq4Y7f7vkYnVOelNLYzdKc28>

gcd(a, b)

if b == 0

return a

return gcd(b, a MOD b)

gcd(12, 18) = gcd(18, 12) = gcd(12, 6) = gcd(6, 0)

Complexity of Fermat

Fermat(n, k)

for i ← 1 to k // O(k)

if powermod(a, n-1, n) !=1 O(log n)

return false

end  
 end  
 return true (probably)

end

O(k log n)

State the performance of Quicksort given data

O(n2) pivot = x[RIGHT]

pivot = (x[L]+x[R])/2 O(n log n)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

pivot = (x[L]+x[R])/2 O(n2)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 5 | 7 | 8 | 6 | 4 | 2 |

insertionsort

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 8 | 2 | 3 | 7 | 6 | 5 | 4 |
| 1 | 8 |  |  |  |  |  |  |
| 1 | 2 | 8 |  |  |  |  |  |
| 1 | 2 | 3 | 8 |  |  |  |  |
| 1 | 2 | 3 | 7 | 8 |  |  |  |
| 1 | 2 | 3 | 6 | 7 | 8 |  |  |
|  |  |  |  |  |  |  |  |

HW: show each pass of the insertion sort

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 9 | 6 | 8 | 6 | 2 | 4 |
| 1 | 3 | 9 | 6 | 8 | 6 | 2 | 4 |
| 1 | 3 | 6 | 9 | 8 | 6 | 2 | 4 |
| 1 | 3 | 6 | 8 | 9 | 6 | 2 | 4 |
| 1 | 3 | 6 | 6 | 8 | 9 | 2 | 4 |
| 1 | 2 | 3 | 6 | 6 | 8 | 9 | 4 |
| 1 | 2 | 3 | 4 | 6 | 6 | 8 | 9 |
| 1 | 2 | 3 | 4 | 6 | 6 | 8 | 9 |
|  |  |  |  |  |  |  |  |

Quicksort

pivot = (x[L] + x[R]) / 2

quicksort(x, 0, 7)

L R

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 9 | 6 | 8 | 6 | 2 | 4 |

pivot = 2.5 = 2

list is x

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 9 | 6 | 8 | 6 | 2 | 4 |
| i | i |  |  |  |  | j | j |
| L | 2 | ij | j | j | j | 3 | R |

quicksort(x,L, i-1)

quicksort(x, i, R)

HW: Do the first pass of quicksort. Show the partition and show which two calls to quicksort would happen next

pivot = (x[L] + x[R]) / 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 5 | 6 | 3 | 4 | 2 | 1 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

fill in the recursive calls….

quicksort(x,  
quicksort(x,

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 5 | 6 | 3 | 7 | 2 | 8 |

show the tree corresponding to the array

for i = 0; i <= n/2 i++)  
 makeheapnode(x, i);

HW: Show makeheap, the first part of heapsort, which turns this array into a maxheap. Use the mapping we discussed in class, and start with n/2 going to down to 0 for the order of the nodes.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 9 | 1 | 6 | 1 | 7 | 3 | 8 | 5 |

HW: Binary search

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 6 | 7 | 7 | 8 | 9 |

show binarySearch(x, 0, 7, 5) (search list x from location 0 to 7 for target = 5)

show binarySearch(x, 0, 7, 7)

Optional HW: Write pseudocode for binarySearch

Golden Mean Search

f(x) = 9 - x2

find x of the max to the nearest .1

goldenMean(f, -2, +3)

L = -2, R = +3

phi = 1.618 = (sqrt(5) + 1) / 2

S = (R - L) / phi = 3.09

a = R - S = 3 - 3.09 = -.09

b = L + S = -2 + 3.09 = + 1.09

R = 1.09

b = -.09

S = (R - L) / phi = 1.909

a = R - S = 1.09 - 1.909 = -.81

L = a = -.81

HW: Golden Mean

f(x) = 14 - (x+1)2 = 13 - x2 - 2x

find x of the max to the nearest .1

goldenMean(f, -3, +5)