General Notes

CS, ML and Stats

Patrick Daly

Contents

1 Computer Science

1.1 Algorithms

DFS Time: O(n), Space: O(n)
Solution exists far away.

Recursive

```
def dfs(node):
  if node:
    # do stuff if pre-order
    if node.left:
      dfs(node.left)
    # do stuff if in-order
    if node.right:
      dfs(node.right)
    # do stuff if post-order
```

Iterative

```
def dfs(node): # if bst, may need to swap search left/right
    visited = set()
    stack = [node]
    while stack:
        current = stack.pop(-1)
        print(current.val)
        if current not in visited:
            visited.add(current)
        if current.left and current.left not in visited:
            stack.append(current.left)
        if current.right and current.right not in visited:
            stack.append(current.right)
    return visited
```

BFS Time: O(n), Space(n)
Solution exists nearby.

Iterative

```
def bfs(node):
```

```
    stack = [node]
    while stack:
        current = stack.pop(-1)
        if current.left:
            stack.append(current.left)
        if current.right:
            stack.append(current.right)
```

Mergesort Time: O(nlogn), Space: O(n)

```
def mergesort(array, start, end):
    if start < end:
        mid = (start+end) // 2
        mergesort(array, start, mid)
        mergesort(array, mid+1, end)
        merge(array, start, mid, end)
```

```
def merge(array, start, mid, end):
    left = array[start: mid+1]
    right = array[mid+1: end+1]
    i, j, k = 0, 0, start
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            array[k] = left[i]
            i += 1
        else:
            array[k] = right[j]
            j += 1
        k += 1
    if j == len(right):
        array[k: end+1] = left[i:]
```

```
def binarysearch(array, val, low, high):
    if high < low:
        # can also return high or low index
        return 'Not found!'
    mid = (low + high) // 2
    if array[mid] > val:
        return binarysearch(array, val, low, mid-1)
```

```
    elif array[mid] < val:
        return binarysearch(array, val, mid+1, high)
    return mid
```

Greatest Common Divisor (GCD) Time: ?, Space: ?

Euclid: Time: O(ln^2min(a, b)), Space: O(1)

```
def gcd(a, b):
    return gcd(b, a % b) if b else a
```

Stein: Time: 60% faster than Euclid in theory, similar in practice Space: O(1)

```
def stein(a, b):
    # binary gcd
    if a == b: return a
    if a == 0: return v
    if b == 0: return a
    if ~a & 1: # a is even
        if b & 1: # b is odd
            return stein(a >> 1, v)
        else: # b is even
            return stein(a >> 1, b >> 1) << 1
    else:
        if ~b & 1: # b is even
            return stein(a, b >> 1)
        else: # b is odd
            if a > b:
                return stein((a-b) >> 1, b)
            else:
                return stein(u, (b-a) >> 1)
```

Fibonacci Time: ?, Space: ?

Iterative

```
def fib_iterative(n):
    if n == 0: return 0
    if n == 1: return 1
    fibs = [0]*(n+1)
    fibs[0] = 0
```

```
    fibs[1] = 1
    for i in range(2, n+1):
        fibs[i] = fibs[i-2] + fibs[i-1]
    return fibs[n]
```

Recursive

```
def fib_recursive(n, dp={0: 0, 1: 1}):
    if n in dp:
        return dp[n]
    return fib_recursive(n-2, dp) + fib_recursive(n-1, dp)
```