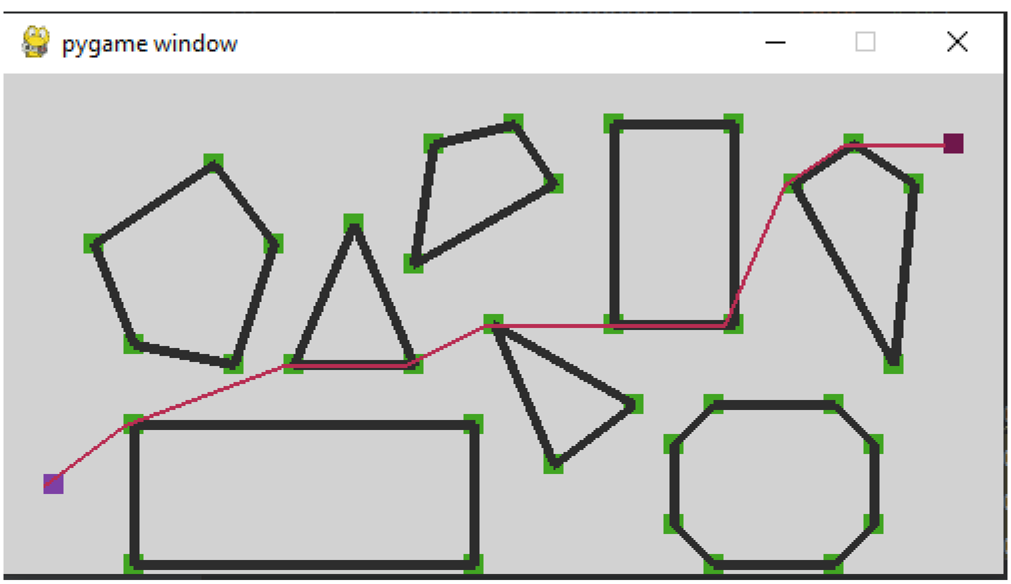
1. N-queens Problem
    a. The lower bound of the function can be set by the simplest of cases where a queen when placed can attack exactly 3 squares. This comes from the fact that a queen when placed has 3-directional and in the lower bound case this because n-3 choices after the first queen is placed. The initial queen still has n choices of placements per column or n * n in the whole state space. With the n choice per column as an initial option we can then multiple the branching factor per queen added. This results in n(n-3)(n-6). Note this is only using the lower bound; not in every n-queen situation.
    b. **Using outside resources from the book**; using the fact that this is a bigpi sequence problem it shows that n(n-3)(n-6) is equivalent to ^3sqrt(n!)
    c. From the product summation equation we can set it as an inequality and test for n to give us a feasible number. N should be in the range (30, 35) as a maximum number. Anything above this should use a better search technique. For instance N = 46 means there are more states than there are atoms in the universe (10^80).
        i. SIDE NOTE: I really don't know how I was supposed to figure this out just with the book; at least the math side of things. Logically sure but to figure out a feasible N that was difficult to find.
2. Compute the order in which states of the above graph are expanded and the returned path for each of the graphs search methods.
    a. DFS
        i. Expanse: Start
        ii. Expanse: Start -> M
        iii. Expanse: M-> Q
        iv. Expanse: Q -> Goal
        v. First Returned Path: Goal <- Q <- M <- Start
        vi. Expanse: Start
        vii. Expanse: Start -> Q
        viii. Expanse: Q -> Goal
        ix. 2nd Returned Path: Goal <- Q <- Start
        x. Expanse: Start
        xi. Expanse: Start -> N
        xii. Expanse: N -> P
        xiii. Expanse: P -> Goal
        xiv. 3rd Returned Path: Goal <- P <- N <- Start
    b. BFS
        i. Expanse: Start
        ii. Expanse: M, Q, N
        iii. Expanse: Q, N
            1. (NOTE: M was taken off but Q was already explored)
        iv. Expanse: Goal, N
        v. Returned Path: Goal <- Q <- M <- Start
        vi. Expanse: N
        vii. Expanse: P
        viii. Expanse: {none left}

c. Uniform Cost Search
    i. Expanse: Start
    ii. Expanse : Start - > N (g(n) = 2)
    iii. Expanse: Start -> M (g(n) = 3)
    iv. Expanse Start -> Q (g(n) = 4)
    v. Expanse: Start -> N -> P (g(n) = 2+4 = 6)
    vi. Expanse: Start -> N -> P -> Q (g(n) = 2+4+1 = 7)
    vii. Expanse : Start -> N -> P -> Goal (g(n) = 2+4+2 = 8)
    viii. UPDATE Goal with g(n) Value
    ix. Expanse: Start -> M -> Q (g(n) 3+4 = 7)
    x. Expanse: Start -> M -> Q -> Goal (g(n) 3 + 4 + 5 = 12)
    xi. DO NOT UPDATE 8<12
    xii. Expanse: Start -> Q -> Goal (g(n) 4 + 5 = 9)
    xiii. DO NOT UPDATE 8<9
    xiv. Start -> N -> P -> Q -> Goal (g(n) 2 + 4 + 1 + 5 = 12)
    xv. DO NOT UPDATE 8<12
    xvi. Best Returned Path: Goal <- P <- N <- Start
d. Greedy Search using Heuristic
    i. Expanse: Start -> Q h=2
    ii. Expanse: Start -> N h=3
    iii. Expanse: N -> P h=1
    iv. Expanse: P -> Q h=2
    v. Expanse: M -> Q h=2
    vi. Expanse: All nodes next to Goal -> Goal (No heuristic val)
    vii. Returned Path: Start -> Q -> Goal (h=2)



Screenshot of resulting path of the aStarAlgo.

It just draws the path returned from the reversed list from my AStarAlgo. It does not show each neighbor being explored just the end path.