Practice Set VIII

Name: Paul Beggs

Problem 1. You are given a list `lst` of distinct integers, `lst` = $[x_0, x_1, \ldots x_{n-1}]$, which has length n which is at least 2.

(a) (4 points) Write (in pseudocode) an algorithm which will return the *second* largest value in the list. For example:

- if `lst` = `[3, 6, 1, 4, 2]` your algorithm should return `4`
- if `lst` = `[-3, -1, -6, -7]` your algorithm should return `-3`

```
def find_second_largest(lst) -> int:
    if len(lst) < 2: # Check length - O(1)
        return None

    # Initial comparison and assignments - O(1)
    if lst[0] > lst[1]:
        maximum, 2nd_max = lst[0], lst[1]
    else:
        maximum, 2nd_max = lst[1], lst[0]

    # Loop through remaining elements - O(n-2)
    for num in lst[2:]:

        # Comparison to find new maximum or second maximum
           O(1)
        if num > maximum:
            2nd_max = maximum
            maximum = num
        elif num > 2nd_max:
            2nd_max = num
    return 2nd_max
```

(b) (2 points) For your algorithm, determinimume a big-Θ estimate for the time complexity, in terms of n.

Solution. $\Theta(n)$

Problem 2. You are given a list `lst` of distinct integers, `lst` = [$x_0, x_1, \ldots x_{n-1}$], which has length n which is at least 2.

(a) (4 points) Write (in pseudocode) an algorithm which will return the largest product of a pair of distinct positions from the list. For example:

- if `lst` = `[3, 6, 1, 4, 2]` your algorithm should return 24
- if `lst` = `[-3, -1, -6, -7]` your algorithm should return 42

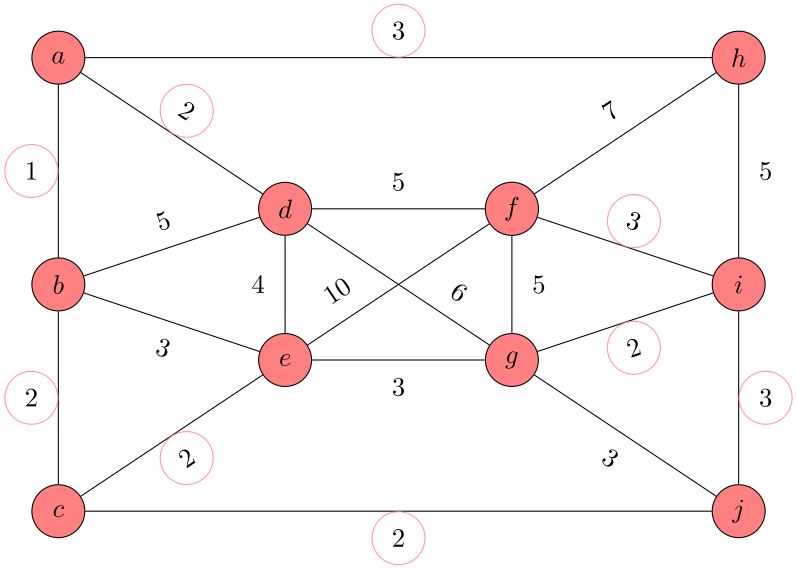
```
def maximum_pair_product(lst) -> int:
    # Initialize the largest and second largest - O(1)
    if lst[0] > lst[1]:
        maximum, 2nd_max = lst[0], lst[1]
    else:
        maximum, 2nd_max = lst[1], lst[0]

    # Initialize the smallest and second smallest - O(1)
    if lst[0] < lst[1]:
        minimum, 2nd_min = lst[0], lst[1]
    else:
        minimum, 2nd_min = lst[1], lst[0]

    # Traverse through the list starting from the third
       element - O(n-2)
    for num in lst[2:]:
        if num > maximum:
            2nd_max = maximum
            maximum = num
        elif num > 2nd_max:
            2nd_max = num

        if num < minimum:
            2nd_min = minimum
            minimum = num
        elif num < 2nd_min:
            2nd_min = num
    return max(maximum * 2nd_max, minimum * 2nd_min)
```

(b) (2 points) For your algorithm, determinimume a big-Θ estimate for the time complexity, in terms of n.

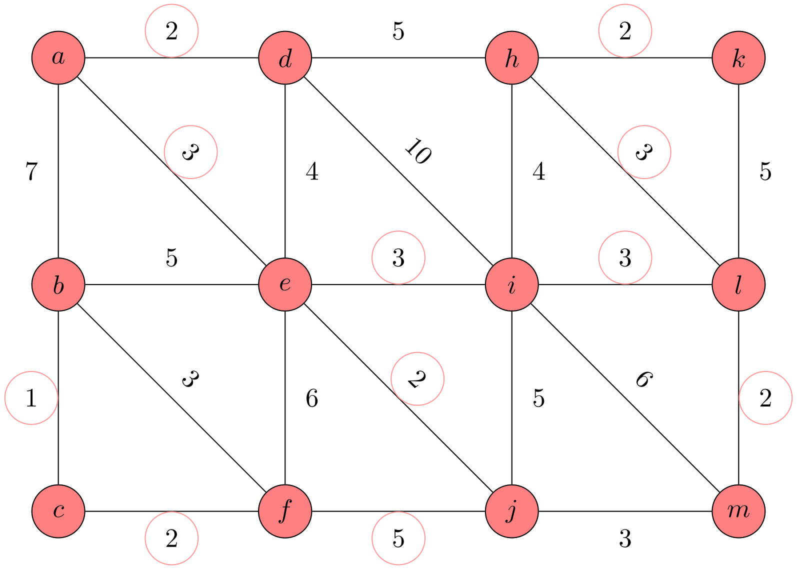
Solution. $\Theta(n)$

Problem 3. (3 points each) For each graph shown, use Prim's Algorithm to determine a minimal weight spanning tree. For each, list the total weight of the tree you find.

(a) $V = \{h, a, d, b, c, e, j, i, g, f\}$; $E = \{3, 2, 1, 2, 2, 2, 3, 2, 3\} = 20$ – Total weight

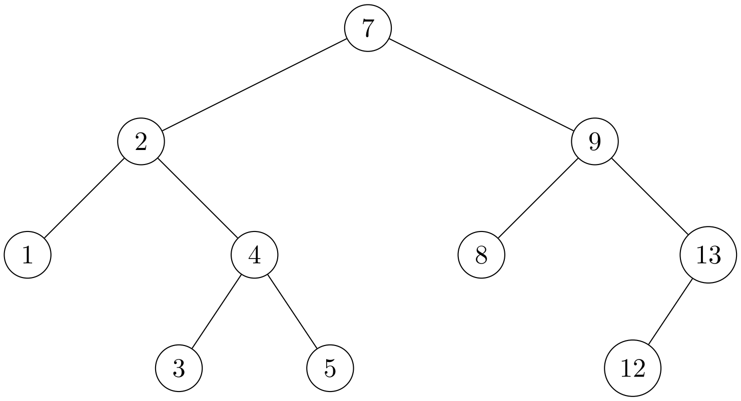


(b) $V = \{b, c, f, j, e, a, d, h, k, l, m\}$; $E = \{1, 2, 5, 2, 3, 3, 2, 3, 3, 2, 2\} = 28$ – Total weight



Problem 4. Consider the list, $[7, 2, 4, 9, 1, 5, 13, 12, 3, 8]$.

(a) (2 points) Construct a binary search tree for this list, using 7 as the root (i.e., do not try to rebalance!)



(b) (2 points) In what order are the nodes visited by a preorder traversal?

> *Solution.* $\{7, 2, 1, 4, 3, 5, 9, 8, 13, 12\}$

(c) (2 points) In what order are the nodes visited by a postorder traversal?

> *Solution.* $\{1, 3, 5, 4, 2, 8, 12, 13, 9, 7\}$

(d) (2 points) In what order are the nodes visited by a inorder traversal?

> *Solution.* $\{1, 2, 3, 4, 5, 7, 8, 9, 12, 13\}$
