CS_MATH395_S17_A2_Brewster_Brandon

Problem 1)

The Clique Problem: A clique, C, in an undirected graph $G = (V, E)$ is a subset of the vertices, C is an element of V, such that every two distinct vertices in C are adjacent. In other words, a clique is a complete subgraph of G. The size of a clique is the number of vertices it contains. The clique problem is the optimization problem of finding a clique of maximum size in a graph.

```
kasan@BB-8:~/Algorithms/ass2/exe$ ./clique
Size of clique to search for: 4
2 3 4 5
kasan@BB-8:~/Algorithms/ass2/exe$ ./clique
Size of clique to search for: 5
No subsets found
kasan@BB-8:~/Algorithms/ass2/exe$
```

Problem 2)

Write computer programs (using C/C++) implementing shellsort, straight insertion sort, selection sort, and quicksort; and compare their performance on random arrays of sizes 10n for n = 2, 3, and 4. Present the comparative performance (as system-time taken by the programs) in a table. Time is reported in milliseconds. A time of zero indicates the operation took less than a millisecond.

```
kasan@BB-8:~/Algorithms/ass2/exe$ ./insertionsort
Shellsort time on size 100: 0
Shellsort time on size 1000: 0
Shellsort time on size 10000: 140.625
kasan@BB-8:~/Algorithms/ass2/exe$ ./quicksort
Shellsort time on size 100: 0
Shellsort time on size 1000: 0
Shellsort time on size 10000: 0
kasan@BB-8:~/Algorithms/ass2/exe$ ./selectionsort
Shellsort time on size 100: 0
Shellsort time on size 1000: 0
Shellsort time on size 10000: 125
kasan@BB-8:~/Algorithms/ass2/exe$ ./shellsort
Shellsort time on size 100: 0
Shellsort time on size 1000: 0
Shellsort time on size 10000: 234.375
```

Problem 3)

a) Write down the adjacency matrix and adjacency lists specifying this graph. (Assume that the matrix rows and columns and vertices in the adjacency lists follow in the alphabetical order of the vertex labels.)

	a	b	c	d	e	f	g
a	0	1	1	1	1	0	0
b	1	0	0	1	0	1	0
c	1	0	0	0	0	0	1
d	1	1	0	0	0	1	0
e	1	0	0	0	0	0	1
f	0	1	0	1	0	0	0
g	0	0	1	0	1	0	0

(a) b, c, d, e

(b) a, d, f

(c) a, g

(d) a, b, f

(e) a, g

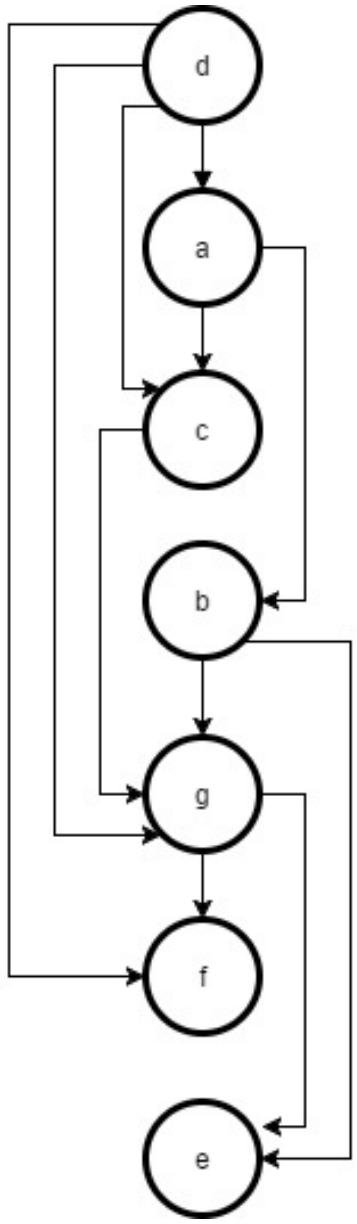
(f) b, d

(g) c, e

Problem 4)

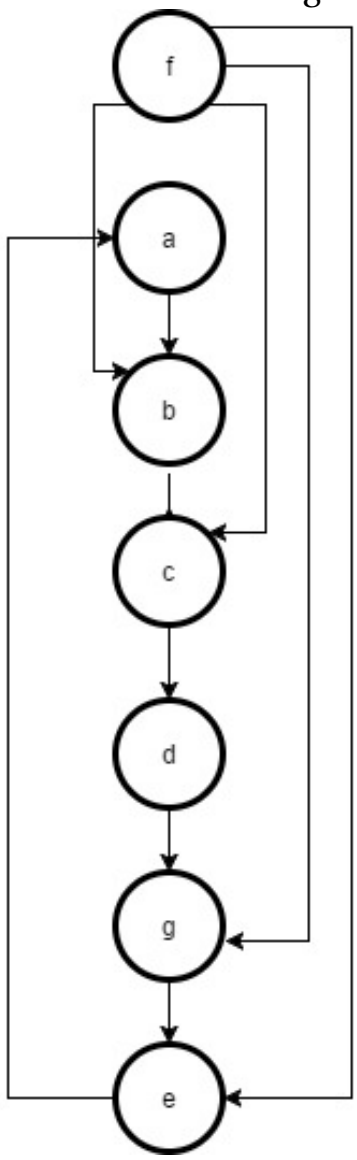
Solve the topological sorting problem for these digraphs applying the DFS based algorithm.

a)



b)

As there is a backedge on this graph, it is not a DAG and cannot be sorted.



Problem 6)

Write pseudocode for a divide-and-conquer algorithm for finding values of both the largest and smallest elements in an array of n numbers

pseudocode:

```
1  int min( vector<int> list )
2  {
3     if( list.size() == 1 ) {
4        return list[0];
5     }
6     vector<int> left;
7     vector<int> right;
8     for( int i=0; i<list.size(); i++ ) {
9        if( i%2 == 0 )
10          left.push_back(list[i]);
11       else
12          right.push_back(list[i]);
13    }
14    int leftmin = min( left );
15    int rightmin = min( right );
16    if( leftmin > rightmin )
17       return rightmin;
18    else
19       return leftmin;
20 }
```

a) Setup and solve (for $n = 2^k$) a recurrence relation for the number of key comparisons made by you algorithm.

$M(n) = M(n-2) + 1$ for $n > 2$
where $M(2) = 1, M(1) = 0$.

b)] How does this algorithm compare with the brute-force algorithm for this problem?

The brute force algorithm will be the same for this, as there will still be the same number of comparisons made. Every value must be checked.