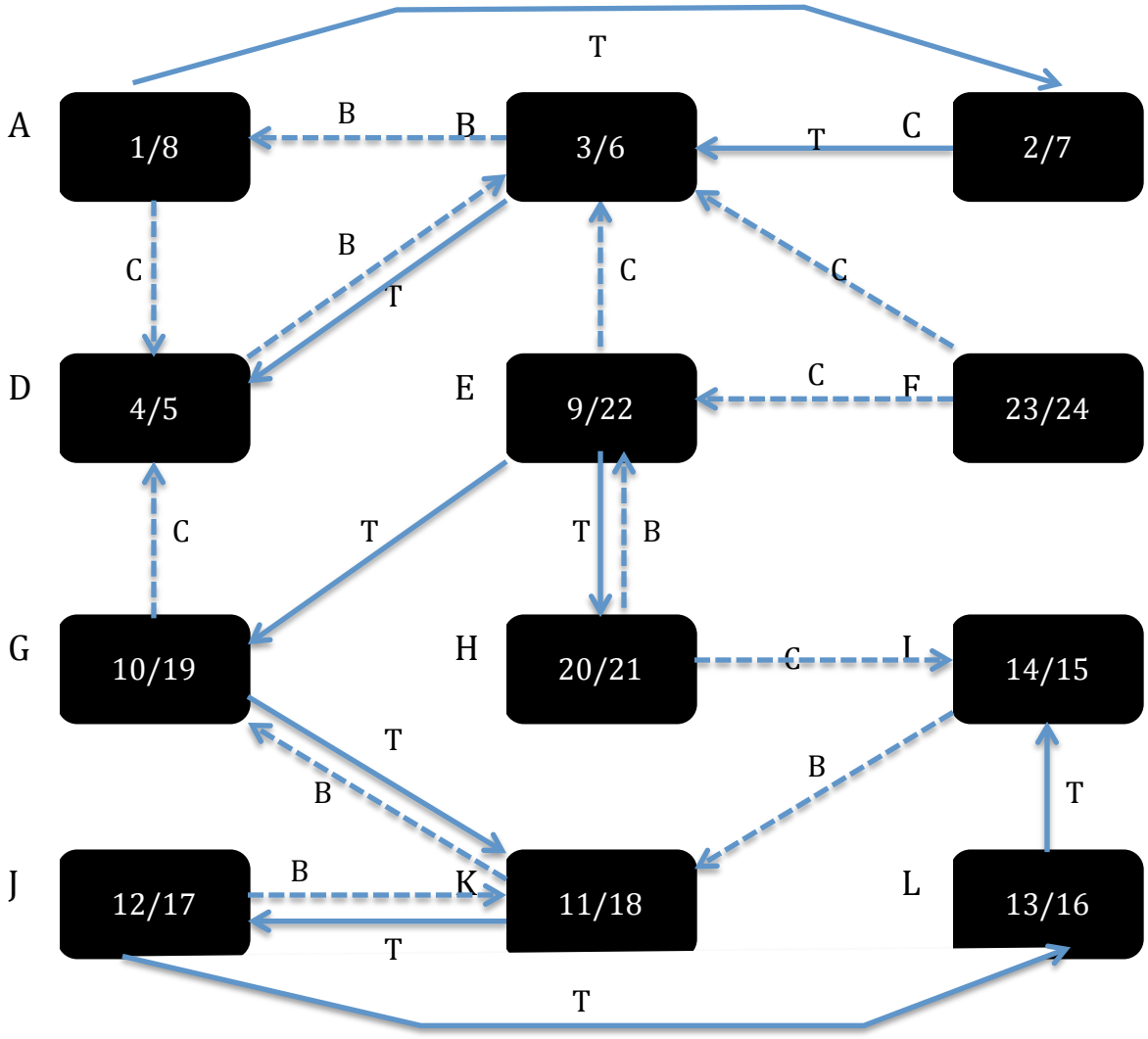
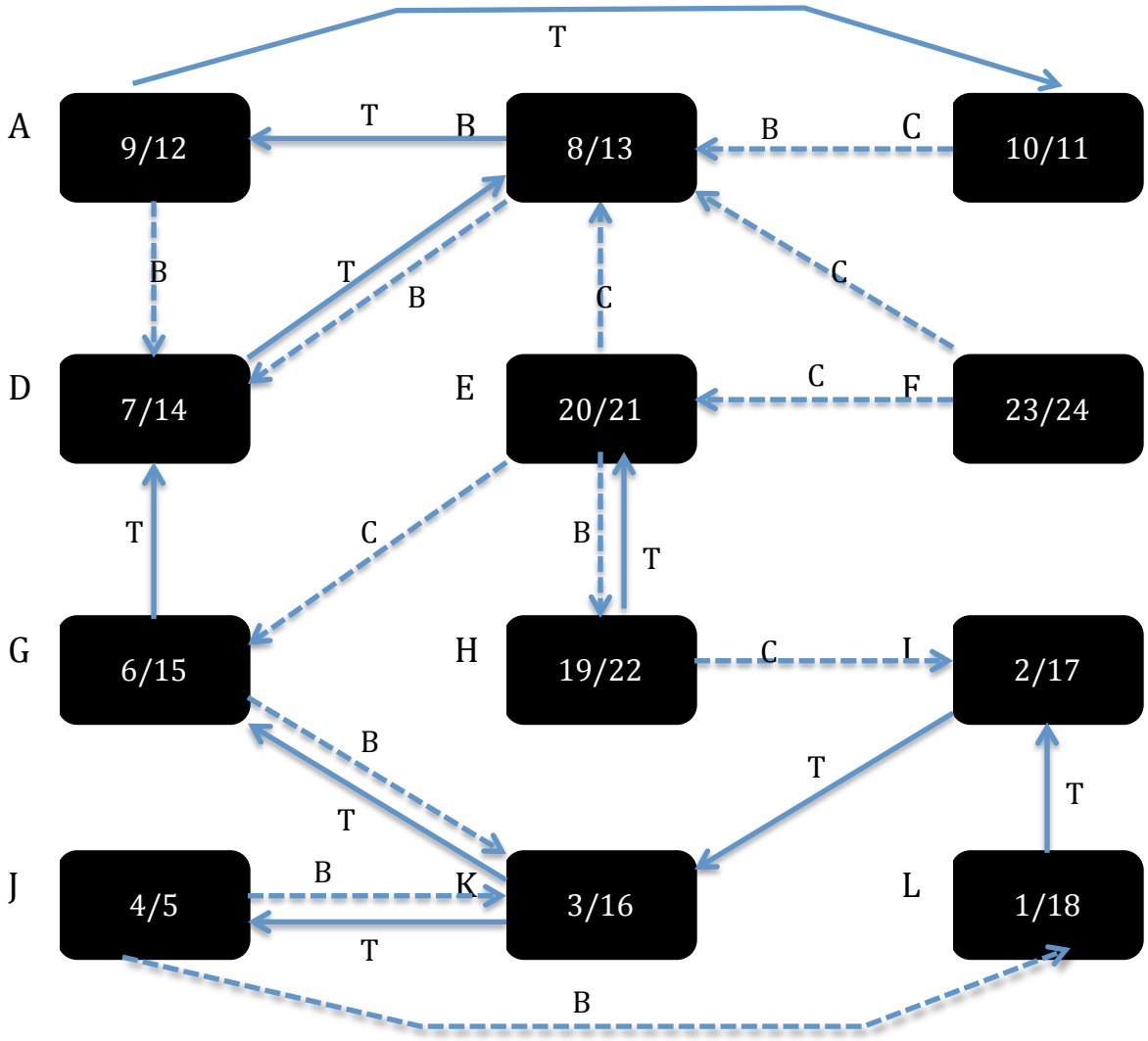
CS600 Web CampusAdvanced Algorithm Design and Implementation

Homework 6

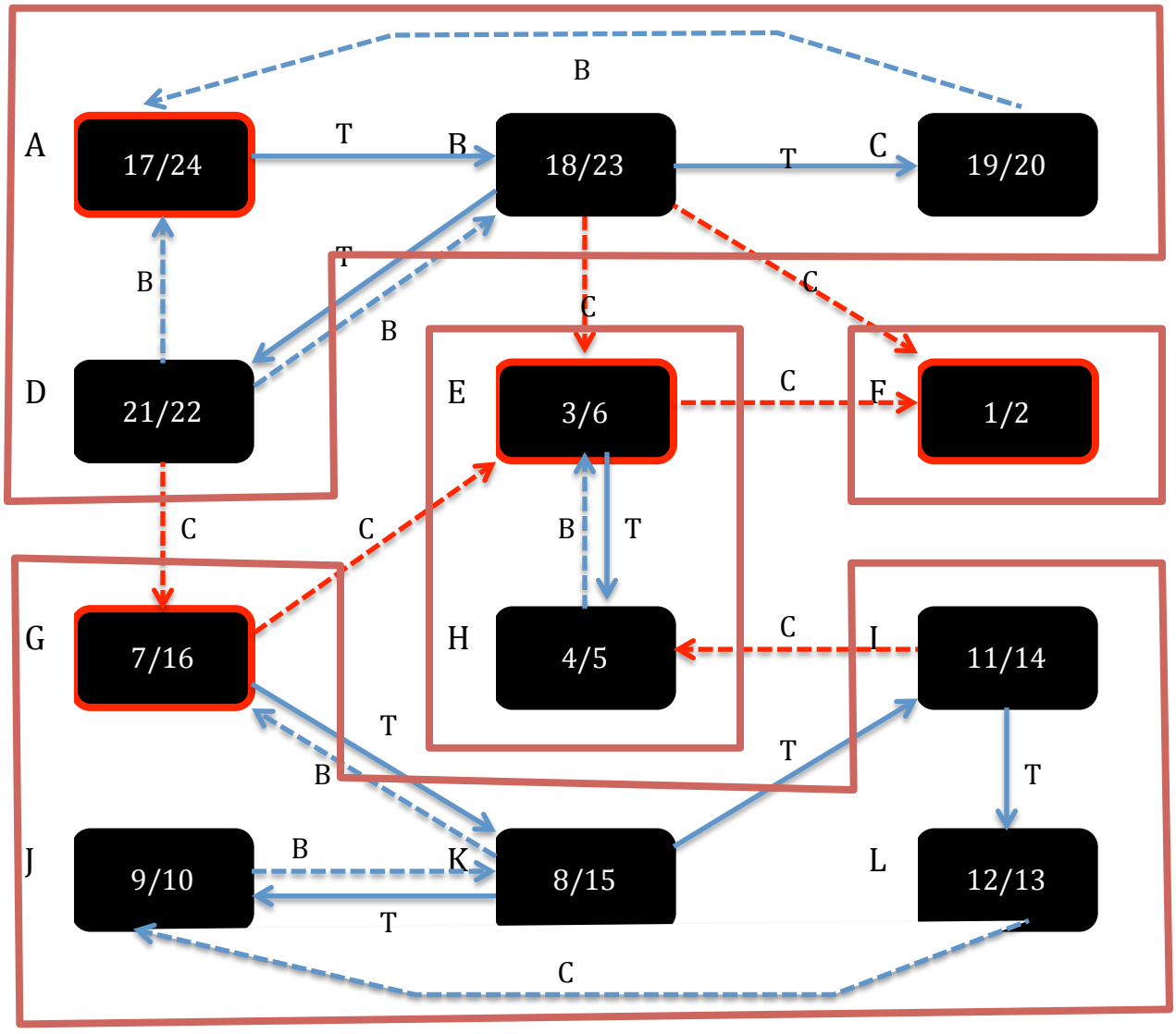
. (a) Perform a DFS (depth-first search) on the given graph. You will visit the nodes in increasing order starting with A (increasing from A to L). Give the discovery u.d and finishing times u.f for every vertex u in the graph. Classify all edges (tree, back, forward, and cross edges).



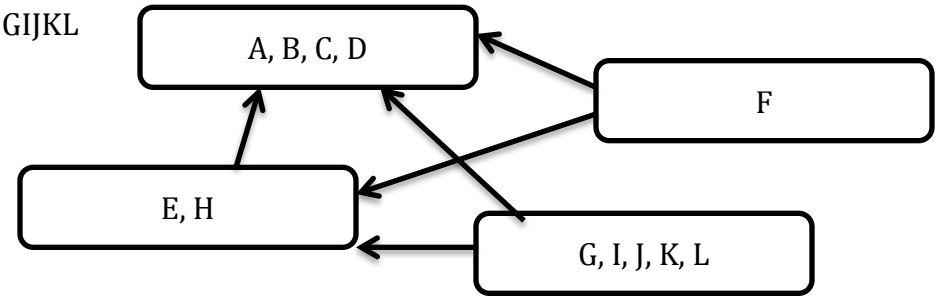
. (b) Perform a DFS (depth-first search) on the given graph. You will visit the nodes in decreasing order starting with L (decreasing from L to A). Give the discovery u.d and finishing times u.f for every vertex u in the graph. Classify all edges (tree, back, forward, and cross edges).



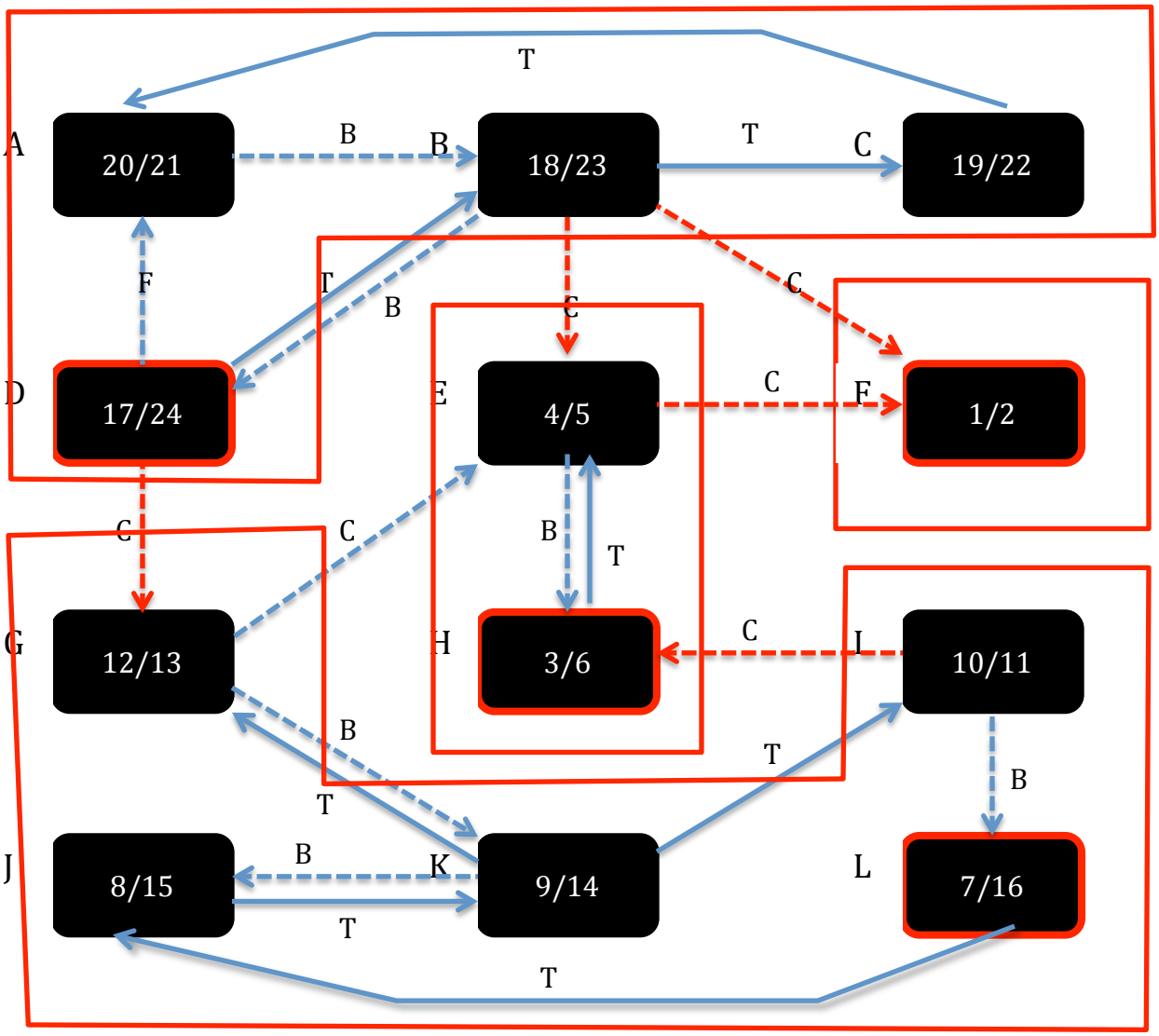
. (c) Determine the strongly connected components (SCC) of the given graph G = (V, E). Use your output of (a) as a starting point for the remainder of the SCC algorithm. Give the discovery and finishing times for the second run of DFS. Give the sets of vertices that form an SCC and draw the corresponding SCC graph.



Vertices sets: ABCD, EH, F, GIJKL
Tree start from: F,E,G,A

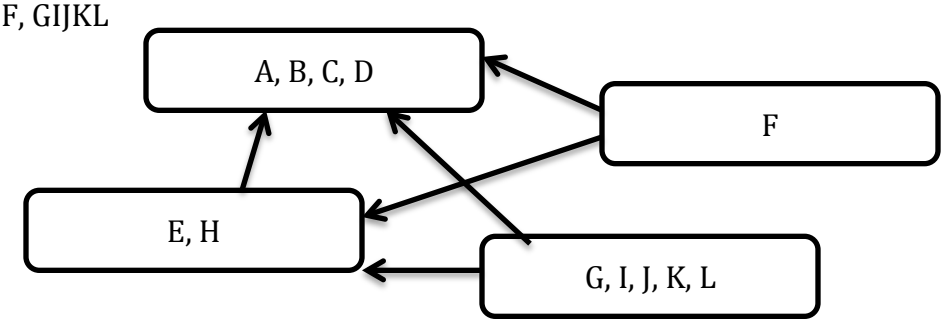


. (d) Determine the strongly connected components (SCC) of the given graph G = (V, E). Use your output of (b) as a starting point for the remainder of the SCC algorithm. Give the discovery and finishing times for the second run of DFS. Give the sets of vertices that form an SCC and draw the corresponding SCC graph.



Vertices sets: ABCD, EH, F, GIJKL

Tree start from: F,H,L,D



2. You need to be able to generate random graphs with certain properties in order to test graph algorithms. In this case you are given a number s of strongly connected components (SCC) the random graph G = (V,E) is supposed to have. For each of the SCC's you are given the number of vertices t_i (with $1 \leq i \leq s$) for this SCC. The graph has therefore $n = \square_{1 \leq i \leq s} t_i$ vertices. Each SCC is connected to at least one other SCC. SCC_1 and SCC_2 are considered connected if there is an edge from one vertex in SCC_1 to a vertex in SCC_2 or an edge from one vertex in SCC_2 to a vertex in SCC_1. Create a random graph using the adjacency matrix representation that satisfies these properties.

Hints:

. How do you make sure that a set of vertices forms a SCC? How many edges do you need to ensure that? Pick the number of edges at random in between the necessary number and the maximum number of edges.

. How do you in combine SCC_1 with vertices {A, B, C, D} and SCC_2 with vertices {E, F, G, H, I} into one graph G given that you use the adjacency matrix representation for G.

. How do you make sure that connecting SCC's does not change the SCC's? You can either do the programming in C/C++ or describe the algorithm in detail using pseudo code (similar to the slides). I suggest that you use pseudo code.

Algorithm to generate a random SCC:
We need at least n edges to ensure a SCC with n vertices. In this case, every vertices is connected one by one and form a circle. We can have at most n(n-1) edges in an n vertices SCC, and there are some rules have to be followed.

In my solution, I generate a circle first and randomly add some other edges except those start from a certain vertices pointing to themselves. Matrix is stored in the array SCC[n][n]

```
SCC[n][n]
Generate_SCC (SCC,n )
{
for i =0 to n-1
    for j = to n-1
        if (i==j) SCC[i][j] = 0
        else if (j==i+1) SCC[i][j] = 1
        else SCC[i][j] = random(0,1)
SCC[n-1][0]=1

/*1<t<n, t is given to extract part of the whole vertices from origin SCC to make a graph, save the adjacent matrix of the t vertices SCC into scc'[t][t]*/
scc'[t][t]=Choose_radom_vertices(t)

return scc'[t][t]
}
```

Algorithm to combine m SCCs:

0	1	Random(0,1)	Random(0,1)	Random(0,1)	Random(0,1)	Random(0,1)	Random(0,1)
1	0	Random(0,1)	Random(0,1)	Random(0,1)	Random(0,1)	Random(0,1)	Random(0,1)
0	0	0	1	Random(0,1)	Random(0,1)	Random(0,1)	Random(0,1)
0	0	Random(0,1)	0	1	Random(0,1)	Random(0,1)	Random(0,1)
0	0	1	Random(0,1)	0	Random(0,1)	Random(0,1)	Random(0,1)
0	0	0	0	0	0	1	Random(0,1)
0	0	0	0	0	Random(0,1)	0	1
0	0	0	0	0	1	Random(0,1)	0

We have m SCCs to generate a random graph.

First of all, I put the first adjacent matrix into the left-up part of the large adjacent matrix of the graph, and then put the next one at the right side but below the first adjacent matrix (as the table shows). In order to ensure that the combination of the SCCs will not change the SCCs, I set all the vacant next to the second matrix to be 0.(let u, v ∈ C,U'~~>v' ∈ C', I want to ensure and suppose there is a path u~~>u' in G. There cannot also be a path v'~~> v in G). After that I can randomly set the rest part to be random(1,0).

Repeat the above process for all the adjacent matrixes of SCCs, and the random graph can be formed.

```
Generate_graph (m)
{
SCC1[t1][ t1],SCC2[t2][ t2],SCC3[t3][ t3],…,SCCm[tm][ tm]
int n=t1+t2+…+tm
Graph[n][n]
int base =0
For i = 1 to m {
    one_counter=0
    for row = o to base+ ti-1 {
        for col= 0 to base+ ti-1 {
            if (base<=row<= base+ti-1 && base<=col<= base+ti-1)
                Graph[row][ col]=SCCi[row-base][col-base]
            If(base!=0&& base<=row<= base+ti-1 && col<=base)
                Graph[row][ col]=0
            If(base!=0&&base<=col<= base+ti-1 && row<=base) {
                Graph[row][ col]=random(0,1)
                If (Graph[row][ col]==1) one_counter++
            }
        }
    }
    if (base!==0 && one_counter==0) {
```

```
        row =random(0, t_i-1)
        col = random(0, base+t_i-1)
        Graph[base+row][ col]=1
    }
    base += t_i
}

return Graph[n][n]
}
```