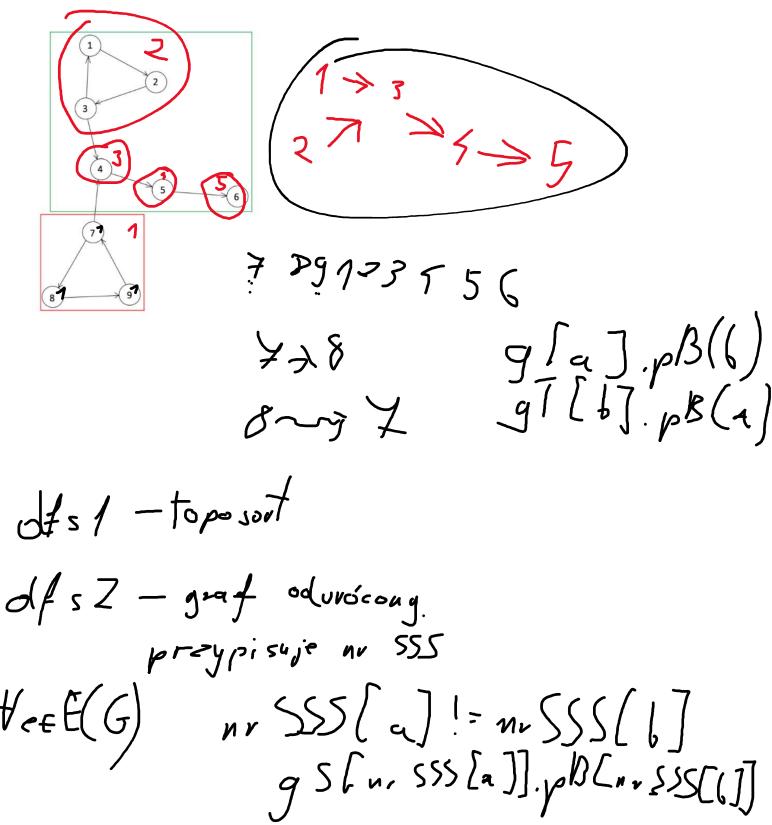
06 December 2024 17:34



```
vector<vector<int>>> g(N);
vector<vector<int>> gT(N);
vector<vector<int>> dag(N);
vector<int> topSort;
vector<int> nrSCC(N, -1);
vector<bool> visited(N);
    visited[s] = true;
         if(!visited[v])
    topSort.push_back(s);
void dfs2(int s, int nr) {
    nrSCC[s] = nr;
     for(int v : gT[s]) {
         if(nrSCC[v] == -1)
         else if(nrSCC[v] < nr) {
   dag[nrSCC[v]].push_back(nr);</pre>
    int n, m, a, b;
cin >> n >> m;
     for(int i = 0;i < m;++i) {</pre>
         g[a].push_back(b);
         gT[b].push_back(a);
         if(!visited[i]) {
     int nr = 1;
for(int i = 1;i <= n;++i) {
    int v = topSort.back();</pre>
         topSort.pop_back();
         if(nrSCC[v] == -1) {
     for(int i = 1;i < nr;++i) {
    cout << "SCC nr: " << i << " zawiera: ";</pre>
              if(nrSCC[j] == i)
    cout << j << " ";</pre>
         cout << endl << "ma polaczenie do: ";</pre>
          for(int v : dag[i])
         cout << endl << endl;</pre>
```

Król zażądał od swojego doradcy, aby sieć była sprawiedliwa, co oznacza, że musi spełniać następującą własność: jeśli z pewnego skrzyżowania v nie da się w żaden sposób dojechać do skrzyżowania u (czyli nie istnieje droga ze skrzyżowania v do skrzyżowania u), to ze skrzyżowania u może istnieć co najwyżej jedna droga do skrzyżowania v. Król uważa, że dzięki temu ludzie mieszkający przy skrzyżowaniu v nie będą z zazdrością patrzeć na ludzi mieszkających przy skrzyżowaniu  $\boldsymbol{u}.$ 

