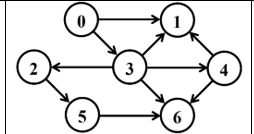
Лабораторная работа 6

**АЛГОРИТМЫ НА ГРАФАХ**

**ЦЕЛЬ РАБОТЫ:** Освоить сущность и программную реализацию: а) способов представления графов; б) алгоритмов поиска в ширину и глубину; в) алгоритма топологической сортировки графов. Разобрать алгоритм Прима и алгоритм Крускала

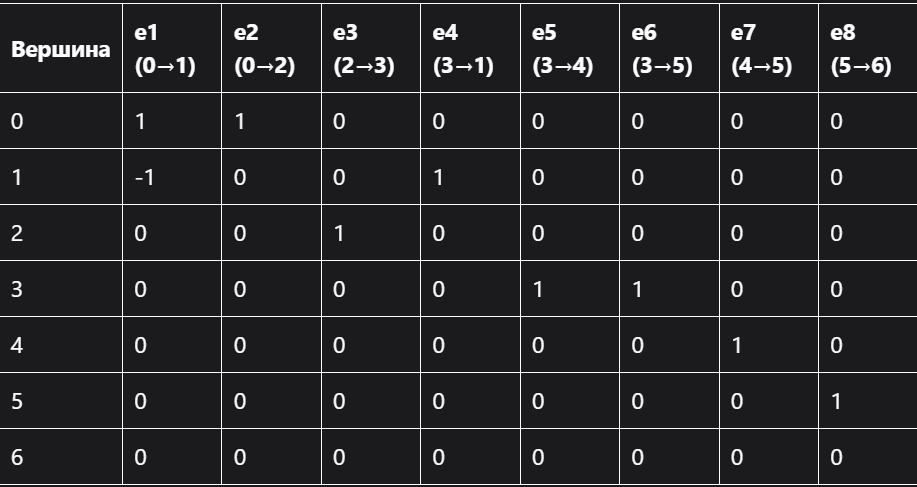
1. Граф:



Матрица смежности:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| gg | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 | 0 | 8 | 0 | 2 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 11 | 0 |
| 3 | 0 | 4 | 9 | 0 | 1 | 0 | 9 |
| 4 | 0 | 3 | 0 | 0 | 0 | 0 | 10 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Матрица инцидентности:



Список вершин:

* 1. 0-3

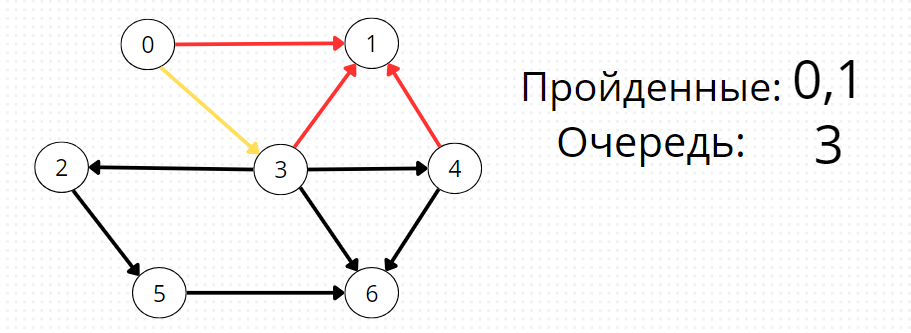
2-5

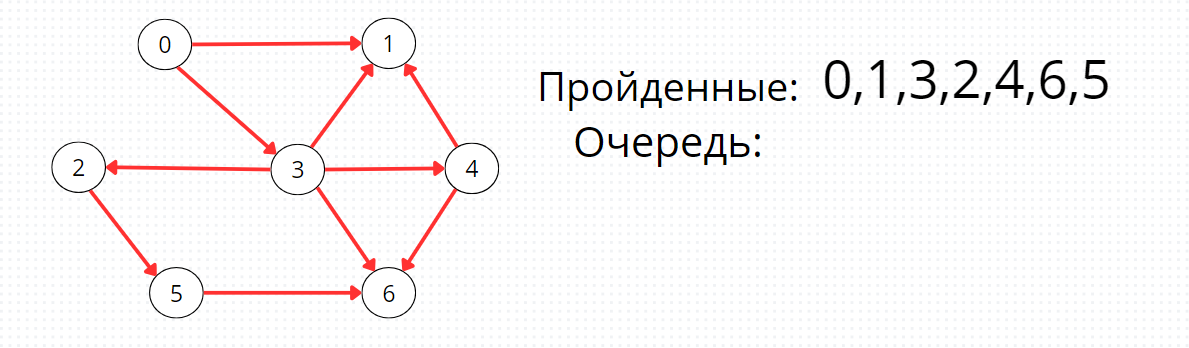
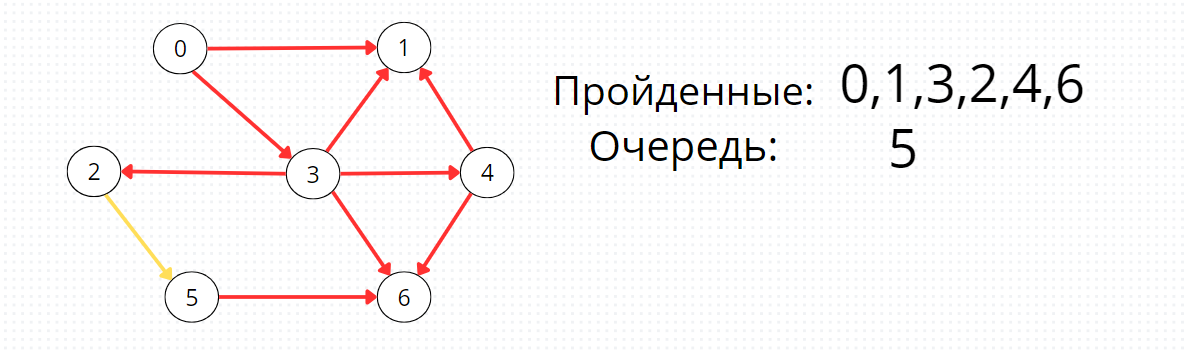
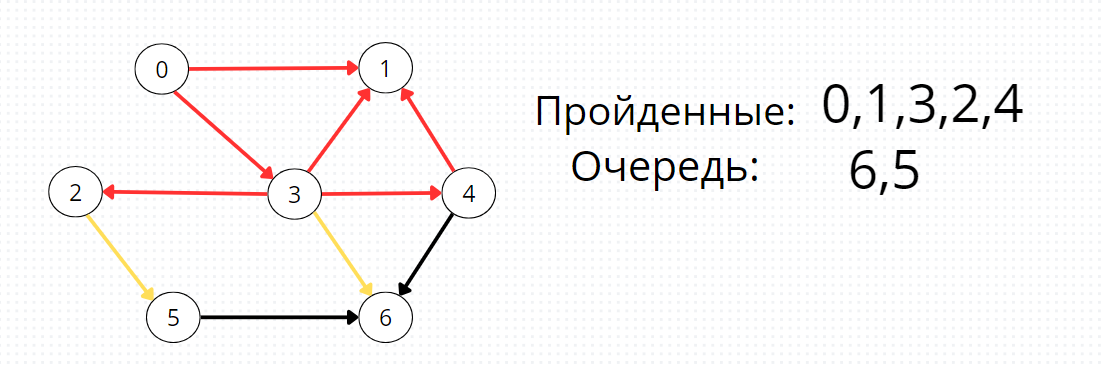
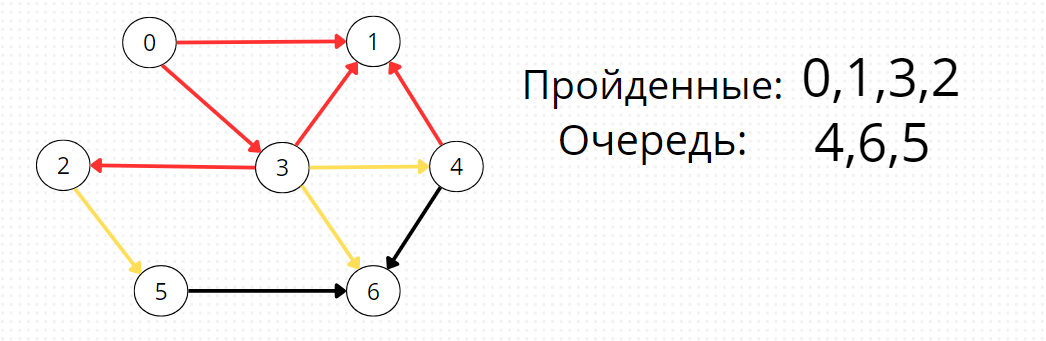
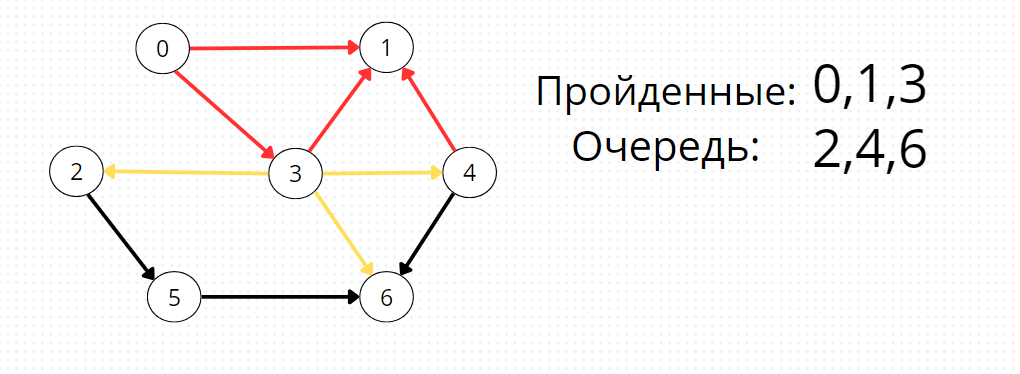
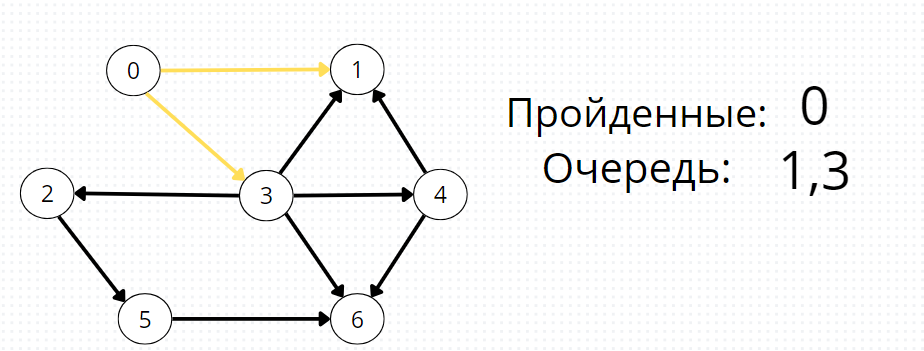
3-1 3-2 3-4 3-6

4-1 4-6

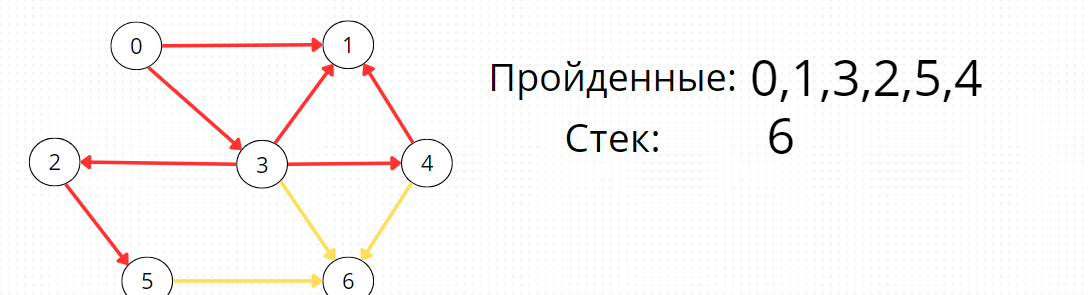
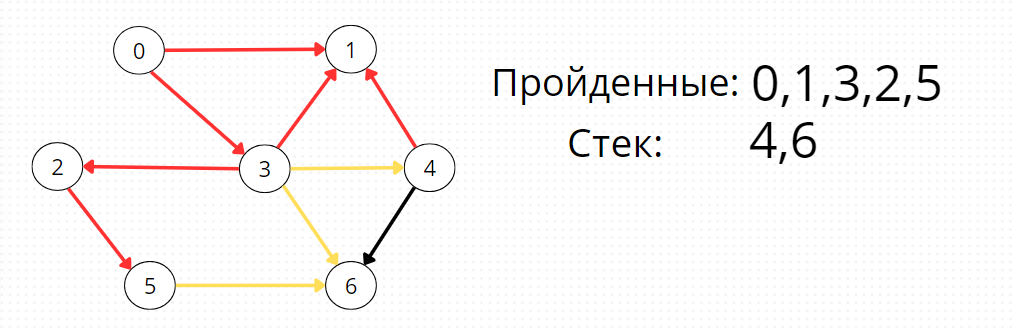
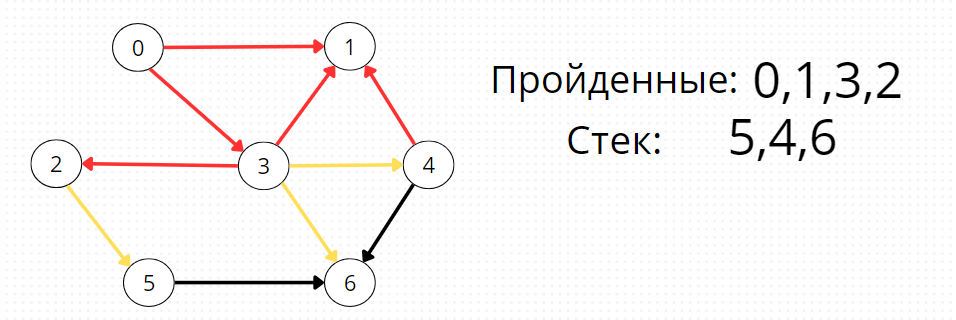
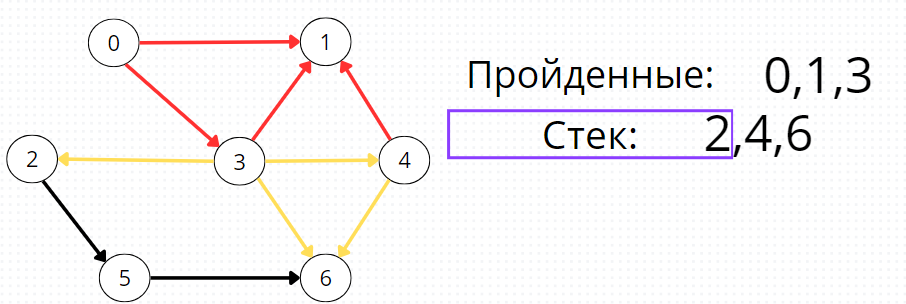
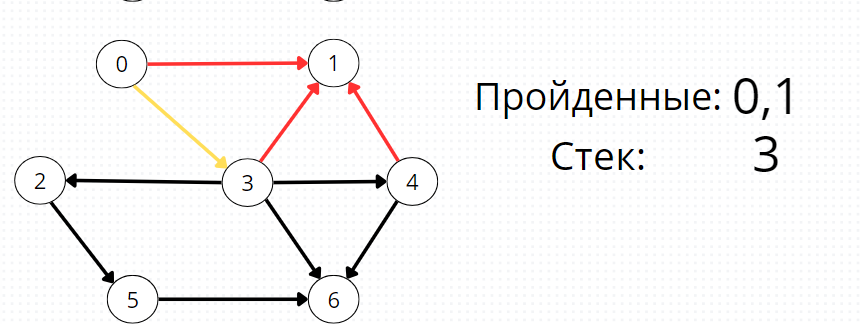
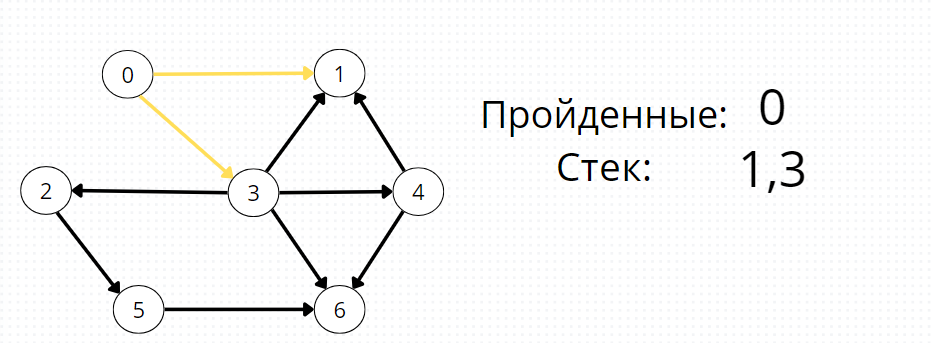
5-6

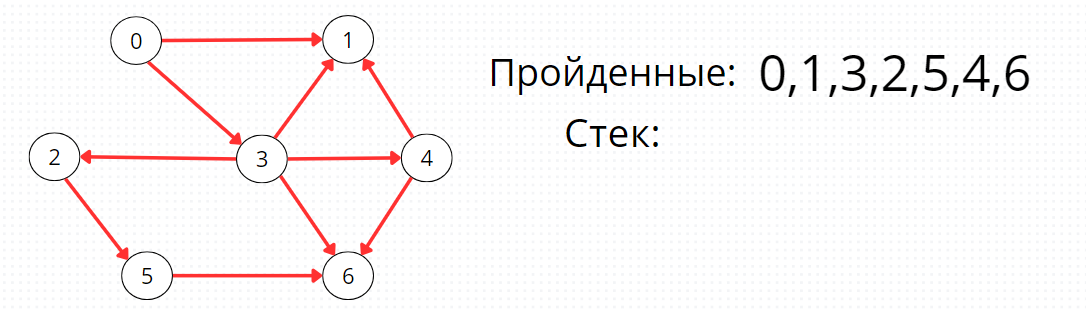
1. BFS:



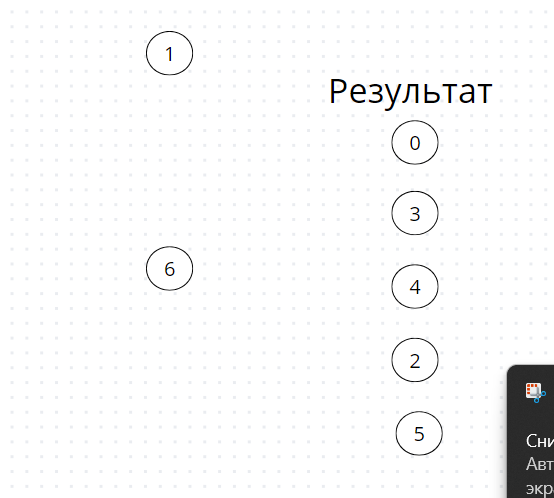
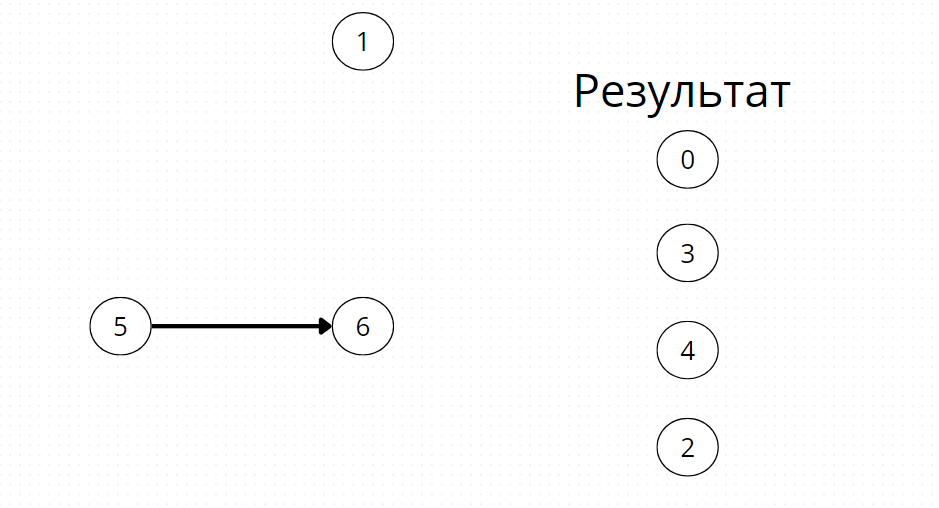
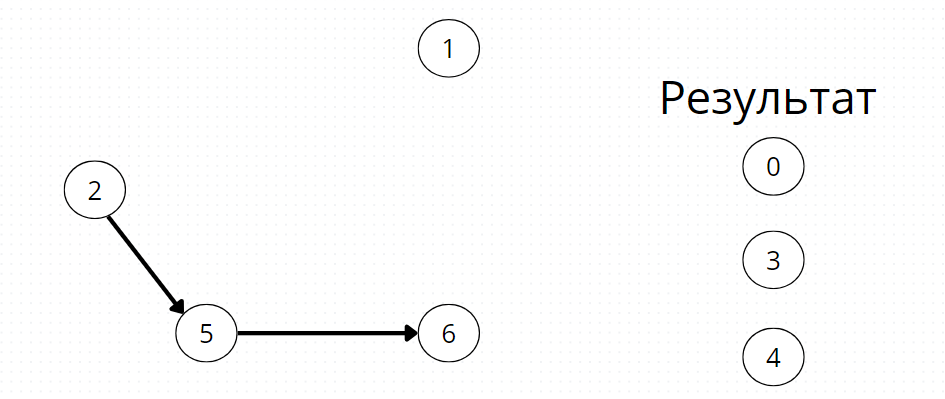
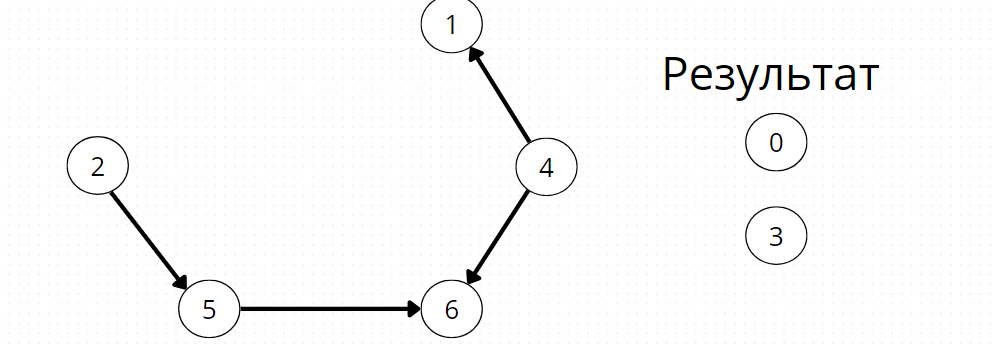
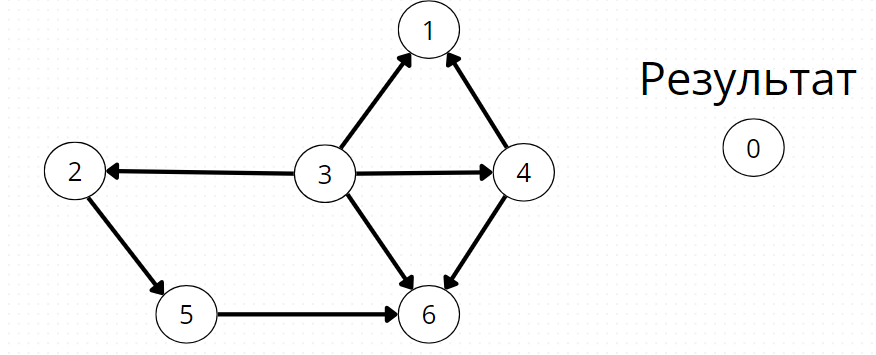
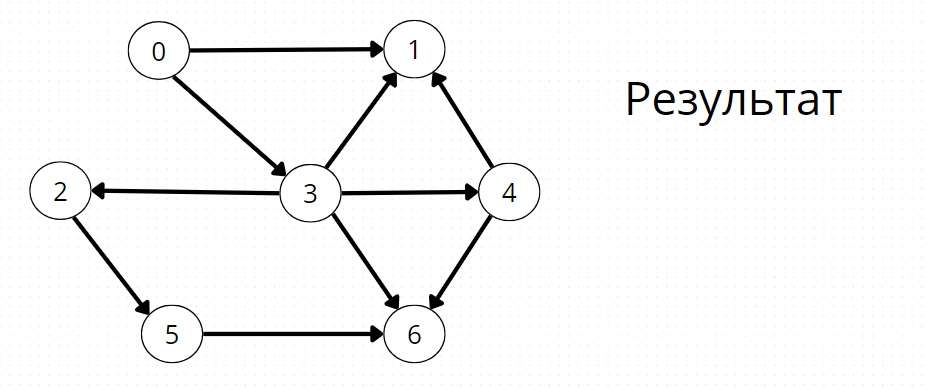


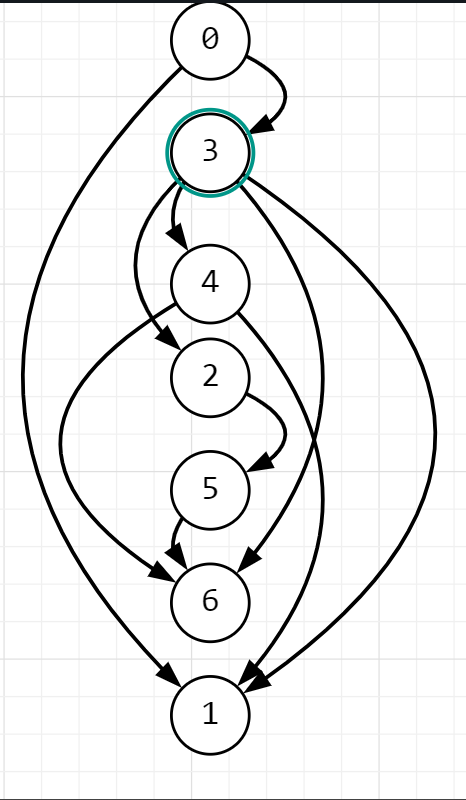
DFS:





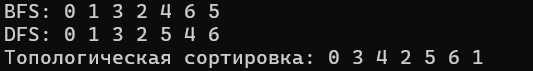
Топологическая сортировка:





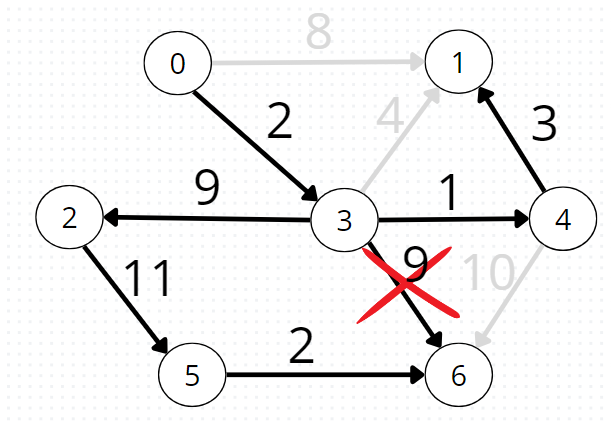
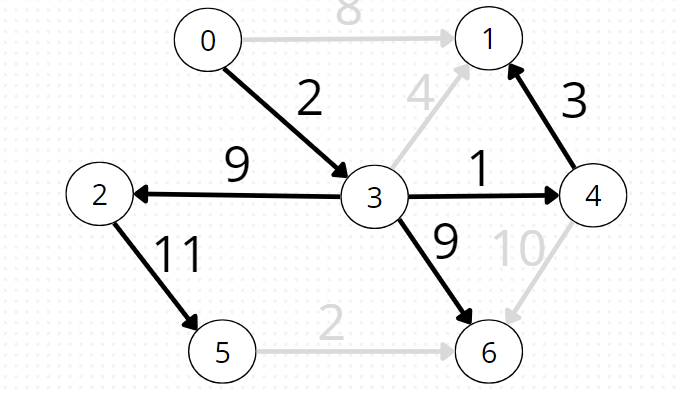
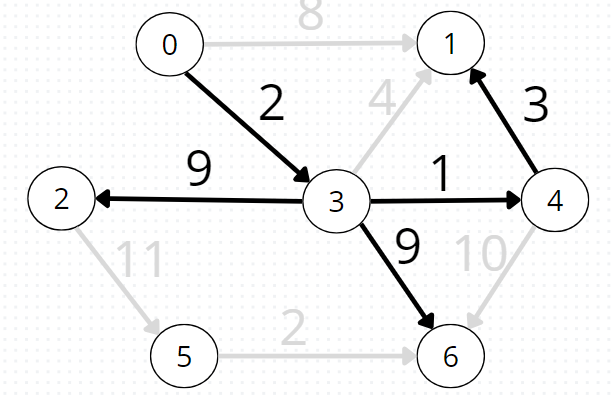
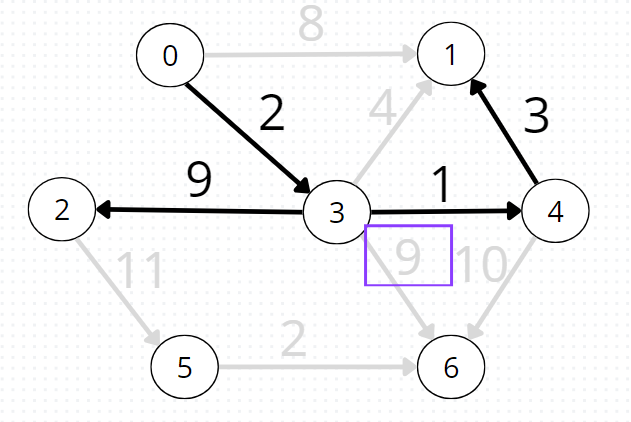
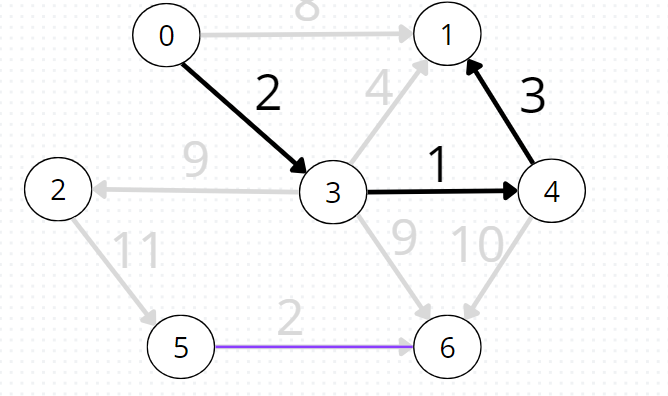
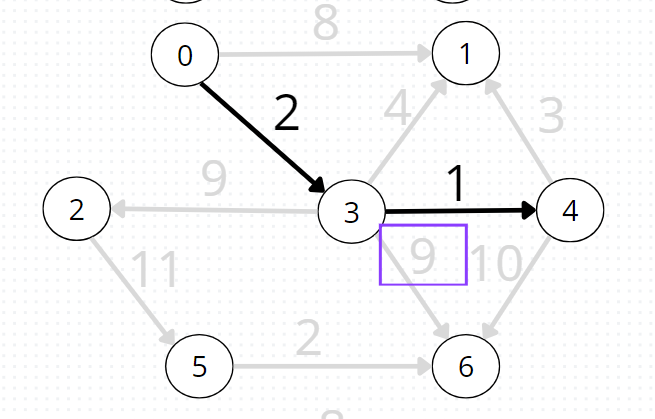
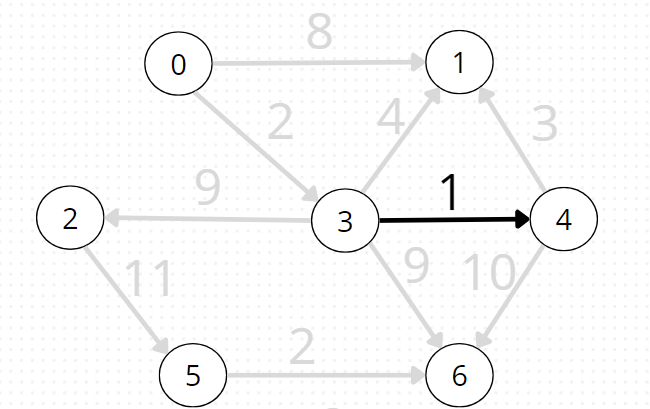
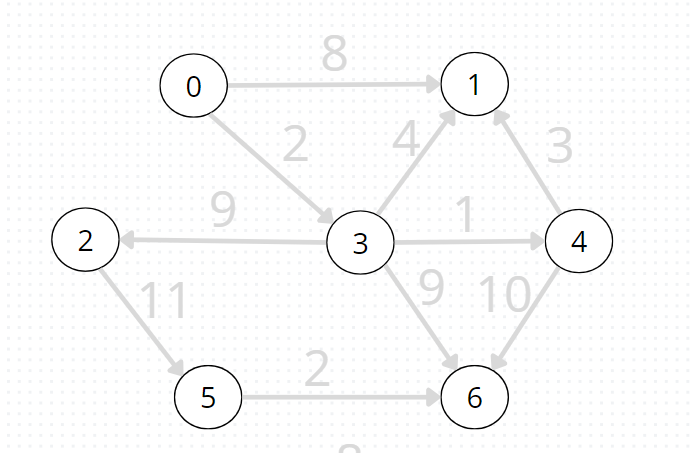
3,4,5:

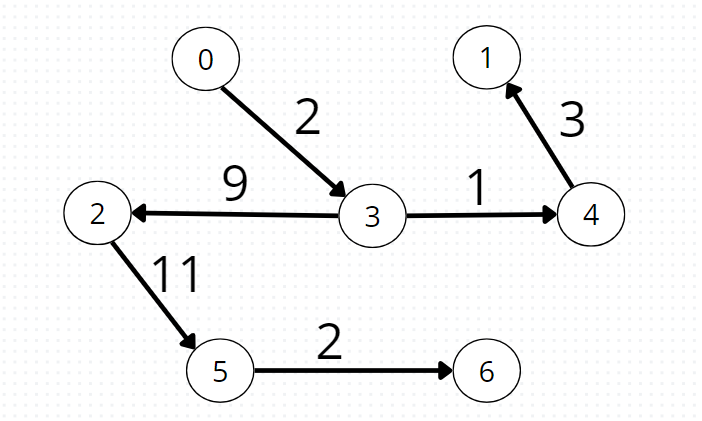
|  |
| --- |
| #include <iostream>  #include <vector>  #include <queue>  #include <stack>  using namespace std;  struct AMatrix  {  vector<vector<bool>> mat;  int vertices;  AMatrix(int n) : vertices(n), mat(n, vector<bool>(n, false)) {}  void addEdge(int u, int v)  {  mat[u][v] = true;  }  void print()  {  cout << "Матрица смежности:\n";  for (int i = 0; i < vertices; ++i)  {  for (int j = 0; j < vertices; ++j)  {  cout << mat[i][j] << " ";  }  cout << "\n";  }  }  };  struct AList  {  vector<vector<int>> adj;  int vertices;  AList(int n) : vertices(n), adj(n) {}  void addEdge(int u, int v)  {  adj[u].push\_back(v);  }  void print()  {  cout << "Список смежности:\n";  for (int i = 0; i < vertices; ++i)  {  cout << i << ": ";  for (int v : adj[i])  {  cout << v << " ";  }  cout << "\n";  }  }  };  AMatrix listToMatrix(const AList& list)  {  AMatrix matrix(list.vertices);  for (int u = 0; u < list.vertices; ++u)  {  for (int v : list.adj[u])  {  matrix.addEdge(u, v);  }  }  return matrix;  }  AList matrixToList(const AMatrix& matrix)  {  AList list(matrix.vertices);  for (int u = 0; u < matrix.vertices; ++u)  {  for (int v = 0; v < matrix.vertices; ++v)  {  if (matrix.mat[u][v])  {  list.addEdge(u, v);  }  }  }  return list;  }  void BFS(const AList& graph, int start)  {  vector<bool> visited(graph.vertices, false);  queue<int> q;  q.push(start);  visited[start] = true;  cout << "BFS: ";  while (!q.empty())  {  int u = q.front();  q.pop();  cout << u << " ";  for (int v : graph.adj[u])  {  if (!visited[v])  {  visited[v] = true;  q.push(v);  }  }  }  cout << "\n";  }  void DFS(const AList& graph, int start)  {  vector<bool> visited(graph.vertices, false);  stack<int> s;  s.push(start);  visited[start] = true;  cout << "DFS: ";  while (!s.empty())  {  int u = s.top();  s.pop();  cout << u << " ";  for (auto it = graph.adj[u].rbegin(); it != graph.adj[u].rend(); ++it)  {  int v = \*it;  if (!visited[v])  {  visited[v] = true;  s.push(v);  }  }  }  cout << "\n";  }  void topologicalSortUtil(const AList& graph, int u, vector<bool>& visited, stack<int>& st)  {  visited[u] = true;  for (int v : graph.adj[u])  {  if (!visited[v])  {  topologicalSortUtil(graph, v, visited, st);  }  }  st.push(u);  }  void topologicalSort(const AList& graph)  {  vector<bool> visited(graph.vertices, false);  stack<int> st;  for (int i = 0; i < graph.vertices; ++i)  {  if (!visited[i])  {  topologicalSortUtil(graph, i, visited, st);  }  }  cout << "Топологическая сортировка: ";  while (!st.empty())  {  cout << st.top() << " ";  st.pop();  }  cout << "\n";  }  int main()  {  setlocale(LC\_ALL, "rus");  AList list(7);  list.addEdge(0, 1);  list.addEdge(0, 3);  list.addEdge(1, 4);  list.addEdge(3, 1);  list.addEdge(3, 2);  list.addEdge(3, 5);  list.addEdge(4, 6);  list.addEdge(5, 2);  list.addEdge(6, 5);  AMatrix matrix = listToMatrix(list);  AList newList = matrixToList(matrix);  list.print();  matrix.print();  newList.print();  BFS(list, 0);  DFS(list, 0);  topologicalSort(list);  return 0;  } |

Пример работы: 

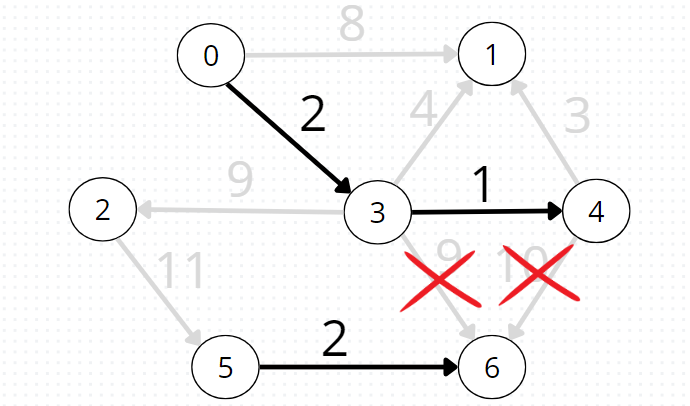
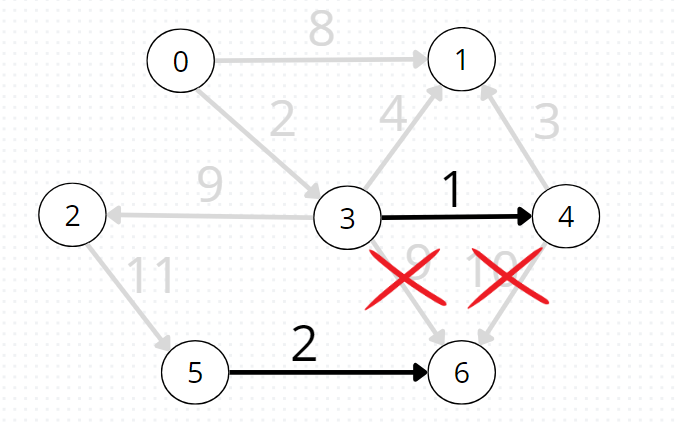
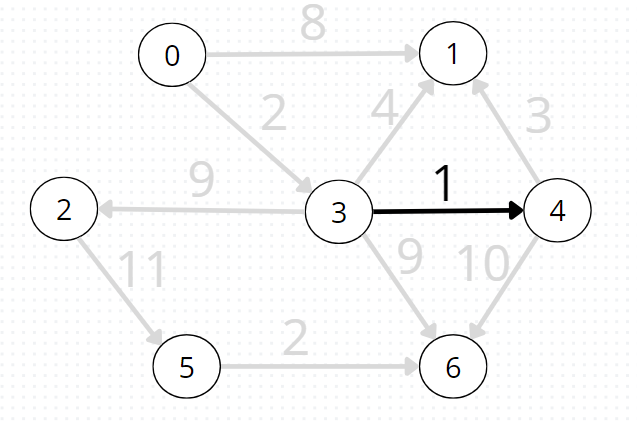
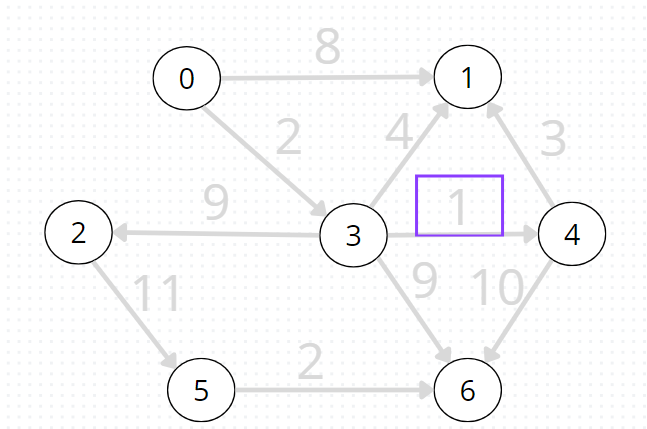
6:

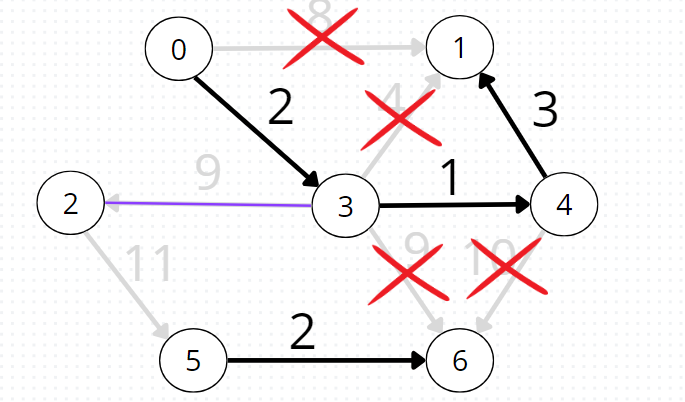
Алгоритм Примма:



Итоговое минимальное остовное дерево 

Алгоритм Крускала:





Итоговое остовное дерево:

