Graph algorithms library

PV264 project

Vojtěch Kaňa, Matej Hulín

The main goal of the proposed project is, as its name suggets, to create
C++ library, which should provide various graph traversal and shortest-paths
algorithms. For implementation, we have chosen following algorithm categories:

Graph traversal

- Depth-first search (DFS) with function prototype:
```
template<typename Graph>
void dfs(Graph& graph);
```

- Breadth-first search (BFS) with function prototype:
```
template<typename Graph>
void bfs(Graph& graph, typename Graph::node_id source);
```

- Iterative deepening depth-first search (IDDFS) with function prototype:
```
template<typename Graph>
bool iterativeDeepening(Graph& graph,
                        typename Graph::node_id source,
                        typename Graph::node_id target,
                        unsigned int maxdepth);
```

- A* with function prototype:
```
template<typename Graph,
         typename Heuristic,
         typename PriorityQueue = BinHeap<typename Graph::node_id>>
void AStar(Graph& graph,
           typename Graph::node_id source,
           typename Graph::node_id target,
           const Heuristic& heuristic);
```
  A* algorithm traverses `graph` from `source` using `heuristic` until it finds
  node `target`.

Single source shortest-paths

- Dijkstra algorithm (using Heap and Fibonacci heap) with function prototype:
```
template<typename Graph,
         typename PriorityQueue = BinHeap<typename Graph::node_id>>
std::enable_if_t<Graph::directedTag && Graph::weighted>
dijkstra(Graph& graph, typename Graph::node_id source);
```

- Bellman-Ford algorithm with function prototypes:
```
template<typename Graph>
std::enable_if_t<Graph::directedTag && Graph::weighted>
bellmanFord(Graph& graph);

template<typename Graph>
std::enable_if_t<!Graph::directedTag && Graph::weighted>
bellmanFord(Graph& graph);
```

- DAG shortest-paths (Topological sort and DFS) with function prototype:
```
template<typename Graph>
std::enable_if_t<Graph::directedTag && Graph::weighted>
dag(Graph& graph);
```

All-pairs shortest-paths

- Floyd-Warshall algorithm with function prototype:
```
template<typename Graph,
         typename Matrix>
std::enable_if_t<Graph::directedTag && Graph::weighted, Matrix>
floydWarshall(Graph& graph);
```

- Johnson's algorithm with function prototype
```
template<typename Graph,
         typename Matrix>
std::enable_if_t<Graph::directedTag && Graph::weighted, Matrix>
johnson(Graph& graph);
```