Lab. 8 - Stochastic Simulation

1. Mont -Carlo Simulation of a Simple Queue System

Study the simple queue Monte Carlo simulation, discussed in the theory class, with arrivals following an exponential distribution, an service times following Erlang distributions. Check for the number of rejected clients for different parameters of the distributions.

2. Generation of Distributions - Histograms

Check the generation of probability distributions studied in the theory class, namey Exponential and Erlang, by generating 1000 events and depicting the corresponding histograms. Use several parameters for the distributions, namely

- For the Exponential distribution use the inverse and/or the accept-reject methods.
- For the Erlang distribution use the accept-reject or the sequence of exponentials methods.

Use the following function signatures,

```
def histogram_exponential(mean, accept_method, fname):
def histogram_erlang(k, mean, accept_method, fname):
```

where the extra parameters specify:

- `accept_method`: (boolean) whether to use the accept-reject method or the alternative.
- `fname`: (string) is a file to store the histogram graphics

3. Generate Discrete Distributions

Specify a function, that generates a discrete event with a value in the range 0..n-1, given a list of size n with the relative likelihood of the corresponding outcomes. Use signature

```
def random_likely(Likelihoods):
```

Note: the likelihoods are relative and their sum may not add up to 1.0.

4. TSP – Selection of Nodes

Adapt the specification of the TSP problem discussed in the Theory namely to use a different node selection method. As discussed, specify 2 alternatives:

```
def select_closer_index(x, ToVisit, AdjMat)
```

that returns the index, in list `ToVisit`, of the node that is closer to `x`, in the adjacency matrix `AdjMat`. Alternatively,

```
def select_likely_index(x, ToVisit, AdjMat)
```

returns the index, in list `ToVisit`, of a node selected with a likelihood inversely proportional to its distance from node `x`, in the adjacency matrix `AdjMat`. For example, if nodes `p` and `q` have distances **3** and **6** from node `x`, then `p` is selected with twice the probability of node `q`.

5. TSP – Assessment

Assess the results obtained with the 3 different alternative methods for node selection in the graph exemplified in the theory class.

Extend your study with some undirected weighted graphs specified in the zip file graphs.zip, with the following format

```
nn na
ni nj wij
```

where the first line indicates the number `nn` of nodes and the number `na` of arcs, and the subsequent lines specify all the `na` arcs, each by a triple `<ni, nj, wij>` where `ni` and `nj` are the node identifiers and `wij` the weight/cost of the connecting arc.

Note 1: Specify a function that returns the adjacency matrix of graph read from file with name fname. Use signature

```
def read_graph(fname)
```

Note 2: In case there are no arc between nodes `i` and `j`, assume a very large cost `wij'` for that arc.