# Experimental design

|  |  |
| --- | --- |
| **Question** | How does Strassen matrix multiplication compare with the traditional matrix multiplication? |
| **Performance indicator** | Number of operations, CPU time |
| **Factors** | Square matrices of order |
| **Levels** |  |
| **Trials** | 10 per design point |
| **Design points** | Until the CPU time and number of operations of Strassen matrix multiplication or lower than traditional. |
| **Outputs** | Size of matrix, number of operations and CPU computation time for both kind of multiplication |

|  |  |
| --- | --- |
| **Question** | When is Dijkstra’s shortest path better than Floyd’s algorithm for solving all pairs shortest path (APSP), depending on ? |
| **Performance indicator** | CPU time |
| **Factors** | Random weighted directed graph (n, m) |
| **Levels** | = 100 … 1000 increments by 100, = 1000 incrementing by 1000 |
| **Trials** | 15 per design point |
| **Design points** | Iteration over m until the cutoff is found. |
| **Outputs** | The number of nodes (n) and edges (m) when Floyd’s algorithm become better than Dijkstra’s shortest path for every n. |

|  |  |
| --- | --- |
| **Question** | Is the extra cost due to balancing worth the savings in average path length for binary search trees? |
| **Performance indicator** | CPU time |
| **Factors** | Real-world data D |
| **Levels** | D = 50 |
| **Trials** | 1 with a “balanced” binary search tree, 1 with a regular one |
| **Design points** | Full factorial |
| **Outputs** | The time for a “balanced” binary search tree, compared to the time for a regular binary search-tree. |

# Graph Coloring

Pilot experiments done on graph (n,m), with values of and as specified above. Values of exceeding are ignored. Moreover, experiments every 500 up to have been performed for and higher. Random numbers have been generated using a seed of 20. The number of iterations for random graph coloring and SIG algorithm has been set to 100. Are included only the conclusions based on the results, not the data themselves.

The results show that whenever the number of edges reaches 3000, independently of the number of nodes, all three implementations of graph coloring results on a *Segmentation fault*.

From the different experiments run, solution quality goes as follow: greedy < random < SIG. It appears that for low number of nodes (<30) all three algorithms give the same results. Random coloring tends to give the same result as SIG until the number of edges exceeds 500. However, all three algorithms give quite similar results, with a difference between the greedy and the SIG algorithms rarely exceeding 5.