Performance Summary

Stable Matching Problem: The algorithm is $O(N^2)$ where N is the number of men and women.

Interval Scheduling: O(Nlog(N)) performance, where N is the number of intervals input. The work of the algorithm is dominated by sorting intervals by increasing finish time.

Scheduling All Intervals: The N intervals are sorted by start time O(Nlog(N)), and the classrooms are kept in a heap that gives access to the classroom with the earliest finish time on its latest scheduled interval. The algorithm puts the next interval into the element at the top of the heap, or into a new element. The element receiving the next interval is sifted up or down to its correct position in the heap. The total amount of work sifting is also O(Nlog(N)).

Scheduling to Minimize Lateness: The work required is dominated by the sort of the N jobs into increasing deadline order - so O(Nlog(N)) performance.

Dijkstra's Algorithm: The 'simple' version of the algorithm (which does not use a heap) is $O(N^2)$ where N is the number of nodes in the graph. The version that uses a heap is O(Mlog(N)) where M is the number of edges, and N in the number of nodes.

Kruskal's Algorithm: Sorting M edges of an N-node graph is O(Mlog(N)). The amount of work for 2M finds and N-1 merges need not be more than O(Mlog(N)).

Prim's Algorithm: The 'simple' version is $O(N^2)$ where N is the number of nodes in the graph. The variant that uses a heap is O(Mlog(N)) where M is the number of edges.

Disjoint Sets: If the initial number of sets is N and merges are done by 'pointing' a lower height tree at a greater height tree, then M finds and N unions would be done in O(Mlog(N)) time. If path compression is also used, then the amount of work is virtually O(M+N).