Sample Algorithm Description and Algorithm Analysis

The priority-based job scheduling problem can be transformed into a sorting problem, and the selection sort algorithm can be used to put the jobs in non-ascending order by their priority. However, before using the selection sort algorithm, we must get those jobs from a *JobCollection* object and put them into an array.

ALGORITHM *PriorityBasedScheduling*(*J[0..n-1]*)

//Schedule an array of jobs by priority

//Input: An array of n job objects, J

//Output: The jobs in J are sorted in non-ascending order by priority.

```
for i ← 0 to n-2 do
        max ← i
        for j ← i+1 to n-1 do
                if J[j].Priority > J[max].Priority
                        max ←j
        temp ← J[i]
        J[i] ← J[max]
        J[max] ← temp
```

Algorithm analysis:

1. The input parameter n is a parameter that characterises the problem size as when its value varies it will have directly impact on the computation time of the algorithm implementation.
2. The operation J[j].Priority > J[max].Priority is chosen as the basic operation of this algorithm as it will have most influence on the computation time of the algorithm implementation and it will be performed most times among all the operations in this algorithm. In this algorithm, the basic operation has only one occurrence.
3. The number of times the basic operation will be performed in the best, worst and average cases remains the same.
4. The number of times the basic operations will be performed when the input size is n is given by:

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-1-i) = \frac{(n-1)n}{2}$$

5. Thus, the efficiency of the priority-based scheduling algorithm is $\Theta(n^2)$ as C(n) = ½*n^2+1/2*n and the highest order item in this polynomial function is n^2.