**GREEDY ALGORITHMS**

<https://www.topcoder.com/community/competitive-programming/tutorials/greedy-is-good/>

Misc -

<https://math.stackexchange.com/questions/48682/maximization-with-xor-operator>

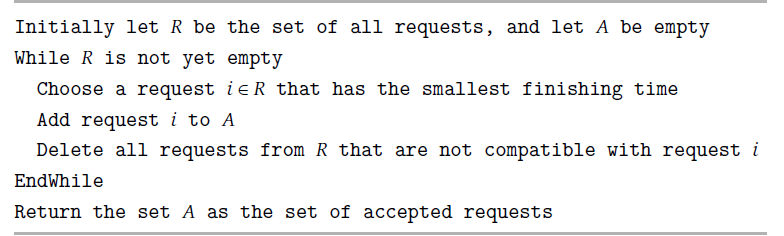
1. **Interval Scheduling -**

Input - Set of requests, each request is a interval with a start time (si) and finish time(fi). Two intervals are compatible if they do not overlap.

Goal - Find out the largest set of compatible intervals.

Solution - Pick the request with minimum finish time first.

Algorithm -



Correctness Argument - Compare with optimal solution according to our greedy strategy.

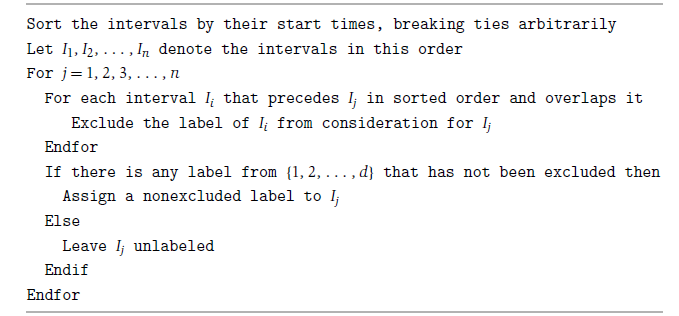
RunTime - O(nlogn)

**2. Interval Partitioning Problem -**

Goal - We have many identical resources available and we wish to schedule all the requests

using as few resources as possible. Analogy - Multiple Classrooms - Resources and Lectures correspond to resources. Scheduling maximum lectures using minimum number of classrooms.

Solution -

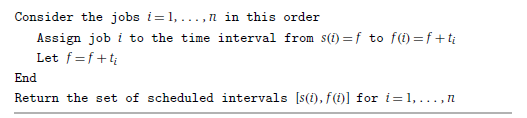


**3. Scheduling to Minimize Lateness:**

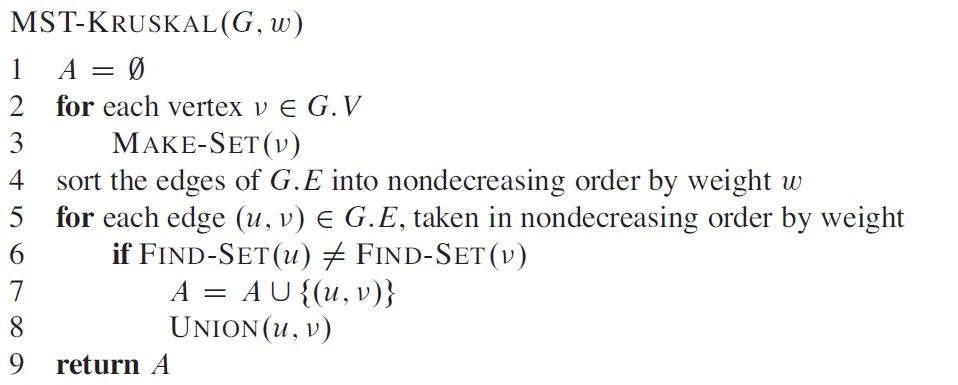
Problem - we have a single resource and a set of n requests to use the resource for an interval of time. Assume that the resource is available starting at time s. The request i has a deadline di, and it requires a contiguous time interval of length ti, but it is willing to be scheduled at any time before the deadline. Each accepted request must be assigned an interval of time of length ti, and different requests must be assigned non overlapping intervals. The goal in our new optimization problem will be to schedule all requests, using non overlapping intervals, so as to minimize the maximum lateness.

Solution -

Order the jobs in order of their deadlines.



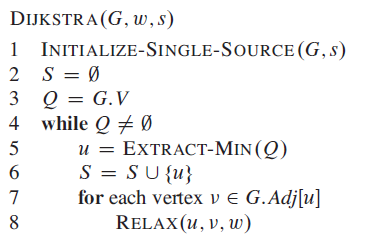
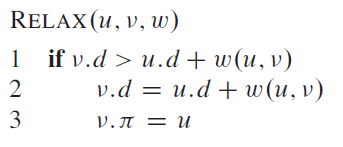
**4. Kruskal’s Algorithm**

****

Runtime - O(E log( V ) )

See - Disjoint Set DS

**5. Dijkstra**



Analysis -

Insert and Extract min is called at most |V| times

Decrease Keys is called |E| times.

Thus the runtime depends on implementation of Min-Priority Queue.

**5. K Centers Problem**

<https://www.geeksforgeeks.org/k-centers-problem-set-1-greedy-approximate-algorithm/>

**6. Connecting N ropes-**

<https://www.geeksforgeeks.org/connect-n-ropes-minimum-cost/>

Simple solution is two sort the ropes by length and then join the ropes with minimum length.

The important thing to note that this kind of repeated sort operations are best done with MIN HEAPS.

**7. Graph Coloring**

The problem is NP- COMPLETE so only approximate solutions exists.

Greedy - provides upper bound - max colors used = d+1 (d = max degree of vertices)

<https://www.geeksforgeeks.org/graph-coloring-set-2-greedy-algorithm/>

**8. Fractional Knapsack Problem**

An **efficient solution** is to use Greedy approach. The basic idea of the greedy approach is to calculate the ratio value/weight for each item and sort the item on basis of this ratio. Then take the item with the highest ratio and add them until we can’t add the next item as a whole and at the end add the next item as much as we can. Which will always be the optimal solution to this problem.