CS218 Programming Assignment

Prerak Contractor

Contents

PART

I

Algorithm

Section 1

Problem

We are given a set of jobs, each with a start time and a end time, with each job requiring one day to finish. Doing any job on a particular day i requires us to pay a cost c_i.

The task is to schedule all the jobs in n days while paying the least cost.

Section 2

Modeling as Graph

Consider a bipartite graph $G(A, B)$ where the set A represents jobs and set B represents days. Edge ab exists if job a can be done on day b. Each vertex b in B is given a weight c_b which is the cost to do a job on day b.

The task then reduces to finding a min weight matching (w.r.t saturated vertices in B) from A to B which saturates all the vertices in A.

Section 3

Pseudo-Code

Algorithm 1 Minimum Weight Matching in $G(A, B)$

Sort Vertices in B based on weights
for vertices $i = 1$ to n in B **do**
    Augment(i)
end for
if All vertices in A present in Matching **then**
    Output: True, Weight of Matching
else
    Output: False
end if

Algorithm 2 Augment Matching for vertex i in B

Run B.F.S starting from vertex i, only visiting vertices which are not fixed, using matched edges to go from A to B while using all edges to go from B to A, stopping when we find vertex in A not in matching.
If such a vertex is found, augment matching by swapping matched edges and edges in path.
Else mark all the vertices visited in B.F.S as fixed

The idea is to greedily add vertices from set B in matching. We first sort all the vertices in B according to their weight and then construct a matching. To do so, starting from the first vertex and moving along the sorted list, we try to find an augmenting path w.r.t the partially constructed matching and augment the matching (using a B.F.S). If for a vertex, it is not possible to do so, we mark all the vertices visited during the B.F.S as fixed, that is, their matchings are fixed and won't be considered for augmentation later.

Section 4

Proof of Correctness

Subsection 4.1

If the algorithm is unable to find an augmenting path for a vertex, then the vertices visited in the corresponding B.F.S will not be part of any other augmentation, and hence can be considered fixed

Assume on the contrary that a vertex b appears in the augmentation later for the first time for some vertex b_0. We know then that before that, all the vertices involved in B.F.S must have been matched to the same set of vertices. Also, in the augmentation path found for b_0, the augmentation path after b will not include any new edges than the ones in B.F.S (since the matched edges are the same, and so are the neighbouring vertices), and hence the same augmentation path would have worked for the unmatched vertex before that.

Hence by contradiction, we can conclude that the vertices would never be part of augmentation again, and hence are fixed.

Subsection 4.2

The algorithm gives an optimal solution

Since at the end, there is no augmenting path for any of the days left out, the matching is a maximal matching, and hence if the algorithm was unable to find a valid matching, there cannot exist another valid matching, and hence the algorithm correctly answers the first question.

Also, if a vertex (day) is not included in the matching, it can only be included by swapping a matched day of lower weight, and hence at any stage of algorithm, it gives optimal matching considering the vertices which have been passed to the augmenting function.

Section 5

Time Complexity

Let n_a represent the number of jobs, and n_b represent the number of days.

Sorting the vertices takes $\mathcal{O}(n_b \log(n_b))$ time. Constructing the graph takes $\mathcal{O}(n_a n_b)$ time. Each B.F.S takes $\mathcal{O}(n_a + n_b + n_a n_b)$ time and augmentation takes $\mathcal{O}(n_a + n_b)$ time. n_b such B.F.S are performed in the algorithm.

Hence, assuming $n_a \approx n_b = n$, time complexity of the algorithm is $\mathcal{O}(n^3)$

However, after applying the optimisation of fixing vertices for failed B.F.S, a boost from 0.8s to 0.07s was observed for largeInput.txt.