Homework 5
Colorado CSCI 5454

Alex Book

September 28, 2021

People I studied with for this homework: Cole Sturza
Other external resources used: N/A

Problem 1

Consider the following algorithm:

Algorithm 1 Maximize-S

```
Input: Graph $G = (V, E)$
Set $S = \{\}$ (empty set)
Define array is_marked of length $|V|$
Set is_marked[$v$] = False for $v = 1, \ldots, n$
for each vertex $v$ do
    if is_marked[$v$] == False then
        DFS-explore-HW5($v$)
    end if
end for
return either $E - S$ or $S$, whichever has larger size
```

Algorithm 2 DFS-explore-HW5

```
Set is_marked[$v$] = True
for each neighbor $w$ of $v$ do
    if is_marked[$w$] == True then
        add edge $(v, w)$ to set $S$
    else
        DFS-explore-HW5($w$)
    end if
end for
```

Correctness

The above algorithm results in two sets of edges, S and $E - S$. S is all "backward" edges, those that create cycles. We know this because to add an edge to S, it must point to a vertex that has already been visited/marked (and thus creates a cycle). $E-S$ is all "forward" edges, those that remain once removing all edges in S (a DAG by definition). These two sets are complements of each other, so returning either one will result in a DAG.

Running Time

The above algorithm has the exact same structure and running time as the given algorithm **DFS-Topo**, which was proven in Lecture 2 notes to have a running time of $O(|V| + |E|)$. Thus, **Maximize-S** also has a running time of $O(|V| + |E|)$ (all changed aspects such as creating S and adding edges to it run in constant time, so the total running time is unchanged).

Approximation Factor

Let OPT be the optimal solution to this problem, and ALG be the solution returned by **Maximize-S**.

The maximum number of edges to keep is the number of edges itself, $|E|$. Therefore, $\text{OPT} \leq |E|$.

The number of edges in ALG is the maximum between $|E-S|$ and $|S|$, or $\max(|E-S|, |S|)$.

We can say the following:

$$\text{OPT} \leq |E - S| + |S| = |E| \tag{1}$$

$$\frac{1}{2}\text{OPT} \leq \frac{1}{2}(|E - S| + |S|) \tag{2}$$

$$\frac{1}{2}\text{OPT} \leq \max(|E - S|, |S|) \tag{3}$$

The transition from line 2 to line 3 is valid because the maximum between two numbers must be of larger magnitude than or equal to the average between those two numbers.

Because ALG is equal to the right side of the above inequality, we can say that ALG is a 2-approximation of OPT.

Problem 2

Consider the following three items:
$v_1 = c, w_1 = c$
$v_2 = c, w_1 = c + 1$
$v_3 = c - 1, w_1 = c$

The capacity is $2c$. The almost-greedy algorithm will look at the items in the order (1, 2, 3). It will put the first item in the knapsack, then fail to fit the second item, then compare items 1 and 2 and return whichever has greater value (they have equal value, so an item of value c will always be returned). However, the optimal solution is using items 1 and 3 for a total value of $2c - 1$.

$$\frac{\text{ALG}}{\text{OPT}} < \frac{1}{2} + \epsilon$$
$$\frac{c}{2c-1} < \frac{1}{2} + \epsilon$$

We can select an arbitrarily large value for c, so we take the limit as $c \to \infty$.

$$\lim_{c \to \infty} \frac{c}{2c-1} < \lim_{c \to \infty} \frac{1}{2} + \epsilon$$
$$\frac{1}{2} < \frac{1}{2} + \epsilon$$

Thus, there exists some $c > 0$ large enough for all $\epsilon > 0$ such that the inequality is satisfied, and the 2-approximation is tight.

Problem 3

Part a

If we remove one edge from OPT (let's call this OPT-Path, as it is then a path not containing any cycles), it is by definition a spanning tree (a subgraph visiting all vertices without cycles). By definition of a minimum spanning tree, OPT-path must have a total weight greater than or equal to the MST (since the MST is the spanning tree with the minimum possible weight for the given graph). Thus, OPT $\geq$ MST.

Part b

As suggested by the hint, we will begin with a "there and back" walk over the MST such that you cross each edge twice, once on the outbound trip, and once again coming back. Such a walk would be a cycle that includes all vertices (as desired) and have a cost of 2·MST. Per what was found in Part a, 2·MST $\leq$ 2·OPT.

Using the information provided (that the provided graph is complete and the edges satisfy the triangle inequality), we can create a cycle shorter than this walk. We do a DFS/pre-order traversal on the MST, connecting nodes in the same order that they are visited. The triangle inequality makes certain that the direct path between two vertices u and v is shorter than or equal to the path from u to v travelling through some vertex w ($(u, v) \leq (u, w) + (w, v)$). So if the MST doesn't contain an edge connecting two vertices that are visited one after another, we can decrease the total weight of the final cycle by connecting those two vertices directly (such an edge must exist because the graph is complete). When the DFS finishes, we simply add the edge connecting the final vertex to the starting vertex. This process will yield a cycle that visits no vertices more than once (aside from the start vertex, which must be visited once at the start and once when connecting to the vertex that is visited last). The cost of this cycle (let's call it ALG) must be less than or equal to the walk described above, as we are essentially taking a "short cut" by directly connecting vertices that are adjacent in the walk if they were not already directly connected. Thus, we can say with certainty that ALG $\leq 2 \cdot$ MST $\leq 2 \cdot$ OPT. Therefore, ALG is a 2-approximation of OPT, the optimal solution to metric TSP.