

# Problem Set 1 Solutions

1. (a) Let  $G = (V, E)$  be a 2-colorable graph.

If  $G$  has several connected components, note that the greedy-algorithm colors each component separately. So a proof for one component will suffice. Hence, without loss of generality assume that  $G$  is connected.

Take any 2-coloring  $\chi^*(\cdot)$  of this graph ( $\chi^* : V \rightarrow \{0, 1\}$ ). Without loss of generality, assume that  $\chi^*(v_1) = 0$ .

The greedy algorithm also assigns  $\chi(v_1) = 0$ . Let  $v_i$  be the first vertex where the coloring functions do not agree, i.e.  $\chi^*(v_i) = c$  and  $\chi(v_i) = c'$ . We will show that no such  $i$  exists.

Just before  $v_i$  is colored by the algorithm,  $\chi$  and  $\chi^*$  agree on  $A$  (this is the first time that they are about to disagree). Therefore,  $\chi$  and  $\chi^*$  agree on  $A \cap \text{Nbhd}(v_k)$ . Since  $\chi^*(v_i) = c$ , all vertices in  $\text{Nbhd}(v_k)$  have the color  $c'$  under  $\chi^*$  because  $\chi^*$  is a valid 2-coloring. Hence all vertices in  $A \cap \text{Nbhd}(v_k)$  have the color  $c'$  under  $\chi^*$ . But  $\chi$  and  $\chi^*$  agree on this set. So the algorithm will have to assign color  $c$  to  $v_i$ . But then  $\chi$  and  $\chi^*$  agree on  $v_i$  which is a contradiction.

Therefore, the algorithm produces a valid 2-coloring.

- (b) See Figure 1.

2. (part a) This is proven by establishing a 2-1 function  $h : \text{OPT} \rightarrow \text{Greedy}$  where OPT (resp. Greedy) denote the jobs scheduled by an optimal schedule (resp. the stated Greedy algorithm). To do this mapping we consider the actual schedules (i.e. where the job starts) in each of OPT and Greedy. Then the proof looks exactly like the proof for the greedy 2-approximation for JISP.

3. (a) Optimal

The key observation that makes Dijkstra's algorithm work is that if the cost of path  $\pi = (s, v_1, v_2, \dots, v_k)$  is  $c_k$ , then for any  $v_{k+1}$ , the cost  $c_{k+1}$  for path  $\pi' = (s, v_1, \dots, v_k, v_{k+1})$ , will always be greater than or equal to  $c_k$ . This condition ensures that a vertex whose cost has been decided by assigning the minimum cost of visiting it from all previously visited vertices, cannot have a smaller cost through any other path.

The observation holds in this case

$$\max_{e \in \pi} c(e) \leq \max_{e \in \pi \cup \{(v_k, v_{k+1})\}} c(e)$$

because the second maximum is over a superset of  $\pi$ .

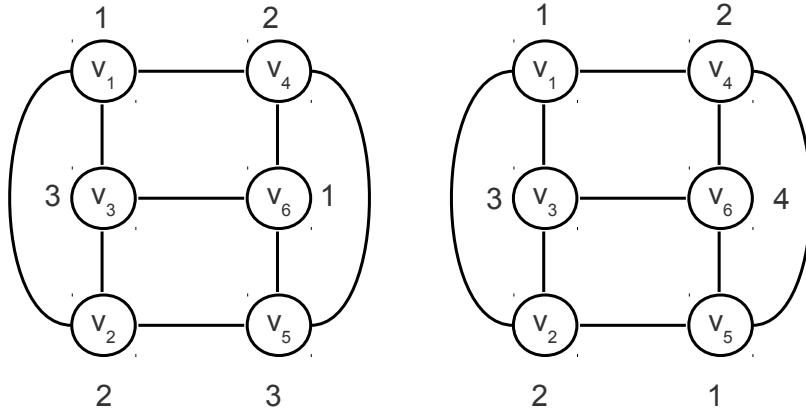


Figure 1: Left : valid 3-coloring , Right : Output of Greedy algorithm

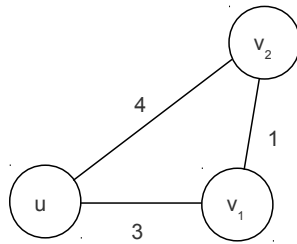


Figure 2: Counter-example for Q3

(b) Not optimal

The observation does not hold if  $w(v_k, v_{k+1})$  is less than  $\min_{e \in \pi} c(e)$ .

$$\min_{e \in \pi} c(e) \not\leq \min_{e \in \pi \cup \{(v_k, v_{k+1})\}} c(e)$$

Figure 2 shows a counter-example. Dijkstra's algorithm will assign cost 3 to  $v_1$ , when the correct cost should be 1.

(c) Not optimal

The observation does not hold-

$$\frac{\sum_{e \in \pi} c(e)}{k} \not\leq \frac{\sum_{e \in \pi \cup \{(v_k, v_{k+1})\}} c(e)}{k+1}$$

if  $w(v_k, v_{k+1})$  is less than the average. Same graph (Figure 2) is a counter-example. Dijkstra's algorithm will assign cost 3 to  $v_1$ , when the correct cost should be 2.5.

4. (a) Semantic array definition:

Let  $V[i, j]$  denote the optimal cost of clustering  $\{x_1, x_2, \dots, x_j\}$  into  $i$  clusters i.e.  $V[i, j]$  is the minimum value of  $\sum_{i'=1}^j \min_{y \in Y} |x_{i'} - y|$  over all choices of  $Y$ , with  $|Y| = i$ .

$V[0, j] = \infty$   $1 \leq j \leq n$  Cannot cluster when  $i = 0$  and  $j \geq 1$

$V[i, j] = 0$   $0 \leq j \leq i \leq k$  If there are at least as many clusters as points then there is no cost as the clusters can be single points.

Then  $V[k, n]$  is the cost of the optimal solution.

- (b)  $V[i, j] = \min_{1 \leq \ell < j} (V[i-1, \ell] + \sum_{h=\ell+1}^j |x_h - x_{\text{median}(i)}|)$  where  $x_{\text{median}(i)}$  is the median of this  $i^{\text{th}}$  cluster  $\{x_{\ell+1}, \dots, x_j\}$

Very informally, if we cluster  $x_1, \dots, x_j$  with  $i$  clusters then the rightmost cluster (which I am calling the  $i^{\text{th}}$  cluster) must contain  $x_j$  and it includes some leftmost point  $x_{\ell+1}$  in the cluster so that the next cluster to the left (the  $(i-1)^{\text{st}}$  cluster) will have  $x_\ell$  as its rightmost point.

- (c) I have not yet thought carefully as to the best implementation of this algorithm. A very rough estimate (and I pretty sure it is an overestimate) is that we have  $kn$  entries and each entry can be computed  $O(n^2)$  steps (by trying each possible  $\ell$  and for each choice of  $\ell$  computing the median for the cluster  $x_{\ell+1}, \dots, x_j$  and the distances to this median. But I am sure this can easily be improved.