CS 331: Algorithms and Complexity (Spring 2024)
Unique Number: 50930, 50935 50940, 50945

Assignment 3 - Solution

Due on Thursday, 8 Febrauary, by 11.59pm

Problem 1: Short Answer Section

(10 pts) True or false. If true, briefly justify, otherwise, provide a counter example . When justifying, restrict answers to no more than a few sentences.

1. **(1 pt)** True, a greedy option picks the best option at each step, without considering the future.

2. **(2 pts)** True, you argue that the differences between the two algorithms are irrelevant to the quality of the solution.

3. **(2 pts)** True, since the shortest path is the path with the fewest edges.

4. **(2 pts)** False, Counterexample:

```
(1, 4), (4, 7), (3, 5), (3, 5)
Try (1, 4), but it has 2 conflicts, so discard it
Try (4, 7), but it has 2 conflicts, so discard it
Pick (3, 5), and since (1, 4) and (4, 7) are discarded, it has only one conflict
Then, our solution yields one job
However, the optimal solution is to pick (1, 4) and (4, 7)
```

5. **(3 pts)** No, the shortest path is not necessarily the path with the fewest edges, Counterexample:

```
(a, b) = 1; (b, c) = 1; (c, d) = 1; (a, d) = 4
Shortest path: a -> b -> c -> d with weight 3
Increment all edges by 1, we get: a -> b -> c -> d with weight 6
However, the shortest path is a -> d with weight 5
```

Problem 2

(10 points) I will denote tasks as (p(i), t(i)) where p(i) is the value of the task and t(i) is the duration of the task.

1. (Smallest duration first) Pick task i that has the minimum duration $t(i)$, or

   Proof. This is not optimal.
   Counterexample:

   ```
   (1, 1), (10, 2)
   Pick (1, 1) first, then (10, 2)
   This yields (1 * 1) + (10 * 3) = 31
   However, the optimal solution is to pick (10, 2) first, then (1, 1)
   This yields (10 * 2) + (1 * 3) = 23
   ```

   □

2. (Most valuable first) Pick task i that has maximum $p(i)$, or

   Proof. This is not optimal.
   Counterexample:

   ```
   (1, 1), (2, 3)
   Pick (2, 3) first, then (1, 1)
   This yields (2 * 3) + (1 * 4) = 10
   However, the optimal solution is to pick (1, 1) first, then (2, 3)
   This yields (1 * 1) + (2 * 4) = 9
   ```

   □

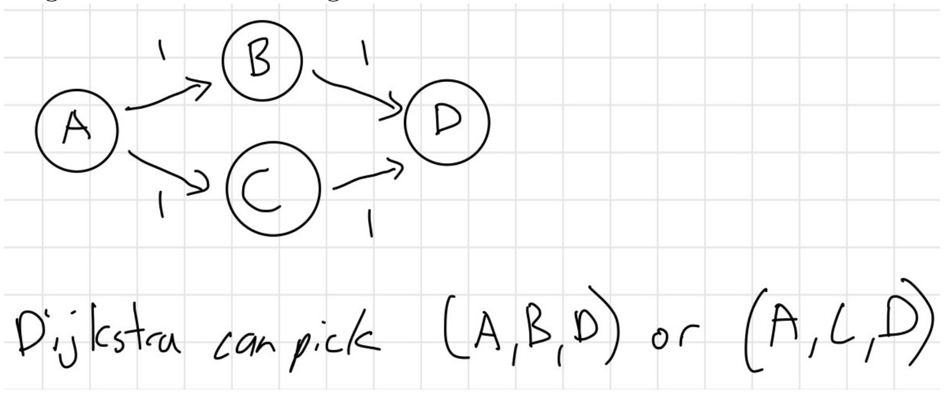
3. (Maximum time-scaled value first) Pick task i that has maximum $p(i)/t(i)$.

   Proof. This is optimal.
   I will prove this using the exchange argument.
   Let the optimal solution $\mathbb{O} = (o_1, o_2, \dots, o_n)$ be the sequence of tasks that minimizes the total value.
   Let the greedy solution $\mathbb{G} = (g_1, g_2, \dots, g_n)$ be the greedy solution where we pick the task with the maximum $p(i)/t(i)$ first.
   Assume there are tasks (o_i, o_j) in $\mathbb{O}$ that is out of order in $\mathbb{G}$.
   We can also assume they're consecutive in $\mathbb{O}$, since we can just iteratively swap consecutive tasks.
   $\therefore$, we have to show the greedy solution doesn't increase the total value of the tasks

after swapping them.
These two tasks have penalty $p(o_i) \cdot t(o_i) + p(o_j) \cdot (t(o_i) + t(o_j))$ in $\mathbb{O}$.
We can expand this to $p(o_i) \cdot t(o_i) + p(o_j) \cdot t(o_i) + p(o_j) \cdot t(o_j)$.
After swapping them, the penalty becomes $p(o_j) \cdot t(o_j) + p(o_i) \cdot (t(o_i) + t(o_j))$.
We can expand this to $p(o_j) \cdot t(o_j) + p(o_i) \cdot t(o_i) + p(o_i) \cdot t(o_j)$.
Now we compare $p(o_i) \cdot t(o_i) + p(o_j) \cdot t(o_i) + p(o_j) \cdot t(o_j)$ ¿ $p(o_j) \cdot t(o_j) + p(o_i) \cdot t(o_i) + p(o_i) \cdot t(o_j)$.
We can subtract $p(o_i) \cdot t(o_i)$ and $p(o_j) \cdot t(o_j)$ from both sides to get $p(o_j) \cdot t(o_i)$ ¿ $p(o_i) \cdot t(o_j)$.
Divide both sides by $t(o_i) \cdot t(o_j)$ to get $\frac{p(o_j)}{t(o_j)}$ ¿ $\frac{p(o_i)}{t(o_i)}$.
Then, by definition of $\mathbb{G}$, $\frac{p(o_j)}{t(o_j)} \geq \frac{p(o_i)}{t(o_i)}$ since o_j was before o_i in the greedy solution.
$\therefore$, the optimal solution $\geq$ the greedy solution.
$\therefore$, the greedy solution is optimal. $\square$

Problem 3

(10 points) Yes, unless the graph has edges of equal weight with equal path weight and length from start to target node.



Dijkstra can pick (A,B,D) or (A,C,D)

Proof. Assume by contradiction that Dijkstra's algorithm picks a different set of edges after doubling the weights of the edges.
Let $\mathbb{D}$ be the set of edges picked by Dijkstra's algorithm before doubling the weights of the edges, we'll denote its weight as W_D.
Let $\mathbb{D}'$ be the set of edges picked by Dijkstra's algorithm after doubling the weights of the edges, we'll denote its weight as $W_{D'}$.
Assume $\mathbb{D}' \neq \mathbb{D}$.
Since Dijkstra's algorithm picks the path with the minimum weight, then $W_{D'} \leq 2 \cdot W_D$.
Then, since $W_{D'}$ is the total weight of the doubled edges, then the total weight of the path before doubling the edges is $W_{D'}/2$.
Using the previous inequality, this implies $W_{D'}/2 \leq W_D$.
Since W_D is the minimum weight of the path, then $W_{D'}/2 = W_D$.
Then, $\mathbb{D}'$ has the same path weight as $\mathbb{D}$.
Now, we show that $\mathbb{D}'$ has the same edges as $\mathbb{D}$.
Assume by contradiction that $\mathbb{D}'$ has a different set of edges than $\mathbb{D}$.

Problem 3 continued on next page...

Let the first differing edge be (u, v) in $\mathbb{D}$.
Then, the weight of the path from the source to u in $\mathbb{D}' = 2 \cdot \mathbb{D}$.
Then, since Dijkstra's algorithm picks the path with the minimum weight, then that means $\mathbb{D}'$ found a cheaper edge.
However, since (u, v) is the cheapest edge in $\mathbb{D}$, then $\mathbb{D}'$ should have picked it since after doubling it retains the quality of being the cheapest .
This is a contradiction, so $\mathbb{D}'$ has the same set of edges as $\mathbb{D}$.
$\therefore$, Dijkstra's algorithm picks the same set of edges after doubling the weights of the edges. $\square$