CPSC 320 2023S2: Quiz 2 Solutions

1 Shortest Path Trees Data Structures (3 points)

Here is the WSP-Greedy-Dijkstra algorithm, almost exactly from the worksheet solutions. Complete three lines of code that construct the parent pointer array $p[1..n]$ of the shortest path tree T rooted at s. This code uses a second array, $p'[1..n]$, that stores and updates "potential parents" during execution of the algorithm. The step of initializing these arrays has already been added to the code below.

```
procedure WSP-Greedy-Dijkstra($s$)
    ▷ create a priority queue whose keys are the initial values of $d[v]$ for each $v \in V$
    Create($Q, n, \infty$)        ▷ set all initial key values in the queue to $\infty$
    ChangeKey($Q, s, 0$)        ▷ change key value $d[s]$ for node $s$ to 0

    ▷ initialize the parent pointers and potential parent pointers to null
    for $i$ from 1 to $n$ do
        $p[i] \leftarrow$ null
        $p'[i] \leftarrow$ null
    $p'[s] \leftarrow s$

    while $T \neq V$ do
        ▷ find a node $v \notin T$ with minimum $d[v]$
        $v \leftarrow$ ExtractMin($Q$)

        ▷ complete the code here which sets the parent pointer of $v$ in $T$
        $p[v] \leftarrow p'[v]$

        for all neighbours $x$ of $v$ do
            if $p[x]$ == null then        ▷ add code here that is true if and only if $x \notin T$
                ▷ update the key for $x$ to be $\min\{d[x], d[v] + w(v,x)\}$)
                ChangeKey($Q$, $x$, $\min\{d[x], d[v] + w(v,x)\}$)

                ▷ update a potential parent here
                if $d[v] + w(v,x) == d[x]$ then $p'[x] \leftarrow v$
```

2 Patchwork Quilt (3 points)

UBC students are putting together a giant $n \times n$ quilt, as a fundraiser for causes important to them. Each participating student is assigned a square $[i, j]$, $1 \leq i, j \leq n$, and can select a pattern for that square from a small set of pattern types. If two adjacent squares (either vertically or horizontally) happen to be of the same type, they belong to the same *patch*.

1. You want to determine the number of patches in the finished quilt. The quilt (with a fixed orientation) is described by an $n \times n$ matrix with position $[i, j]$ denoting the type of the patch at square $[i, j]$ of the quilt. Which graph algorithm from Chapter 3 of the text would be most useful to answer this question?

   ○ Topological ordering in a directed graph
   [✓ Finding connected components in an undirected graph]
   ○ Testing if an undirected graph is connected
   ○ Testing if a directed graph is strongly connected
   ○ Testing bipartiteness

2. What is the runtime of your chosen algorithm, when applied to determine the number of patches in an $n \times n$ quilt? Choose one.

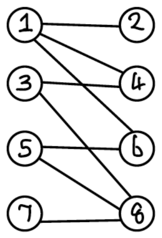
   ○ $\Theta(n)$ ○ $\Theta(n \log n)$ [✓ $\Theta(n^2)$] ○ $\Theta(n^3)$

3 Graph Colouring (5 points)

Recall the graph colouring problem from Assignment 2: You are given an undirected, connected graph $G = (V, E)$. Assume that $V = \{1, 2, \ldots, n\}$. Your goal is to colour the nodes of the graph so that no two adjacent nodes have the same colour, ideally using as few colours as possible. Here is the greedy algorithm from the solution to that assignment, where a node ordering is provided as part of the input:

```
procedure GREEDY-COLOUR($G$, Order[1..$n$])
    ▷ colour the nodes in the order given by Order[1..$n$] using a greedy approach
    colour node Order[1] with colour 1.
    $C \leftarrow 1$
    while not all nodes are coloured do
        move on to the next node, say $v$
        if not all colours between 1 and C are used on the nodes adjacent to $v$ then
            choose the lowest such colour for $v$
        else
            use colour $C + 1$ for $v$, and add 1 to $C$
```

Example: If the graph is as follows, and Order[1..n] is 3,6,5,7,4,1,2,8, then the nodes are coloured as follows, for a total of three colours:



Node order	3	6	5	7	4	1	2	8
Colour	1	1	1	1	2	3	1	2

1. For the graph above, how many colours are used when the input to Greedy-Colour is the Lo-to-Hi ordering 2 7 5 4 3 6 1 8

   ○ 1 ○ 2 [✓ 3] ○ 4

2. For the graph above, how many colours are used when the input to Greedy-Colour is the Hi-to-Lo ordering 1 8 5 3 6 4 7 2

   ○ 1 ○ 2 [✓ 3] ○ 4

3. Give an ordering of the nodes in the above graph which guarantees that GREEDY-COLOUR uses only two colours.

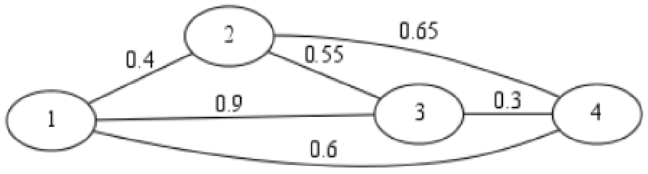
4. Indicate whether you think each of the following statements is true or false, for graphs more generally.

   (a) For *any* graph that is colourable with just two colours, is there always *some* ordering of the nodes of the graph on which Greedy-Colour uses only two colours.
   [✓ True] ○ False

   (b) For any graph that is colourable with just three colours, is there always some ordering of the nodes of the graph on which Greedy-Colour uses only three colours?
   [✓ True] ○ False

4 Max-Min Categorization

Suppose that we change the goal of the photo categorization problem in the clustering worksheet to *maximize the minimum intra-cluster edge similarity.* We call this Max-Min categorization. (In contrast, the goal in the worksheet was to minimize the maximum inter-category edge similarity.) For the 4-node example from the worksheet, shown below, the minimum intra-cluster edge similarity of cluster $\{1, 3, 4\}$ is .3, and the minimum intra-cluster edge similarity of cluster $\{1, 4\}$ is .6. Also, we'll define the intra-cluster edge similarity of any set of size 1 to be 1.



1. For the 4-node example from the worksheet, shown below, and $c = 2$, which of the following are optimal Max-Min categorizations? Check all that apply.

   ○ $\{\{1,2\},\{3,4\}\}$ [✓ $\{\{1,3\},\{2,4\}\}$] ○ $\{\{1,4\},\{2,3\}\}$ ○ $\{\{1,2,3\},\{4\}\}$

2. Here is a greedy algorithm that produces a categorization:

```
function New-Clustering-Algorithm($n, E, c$)
    ▷ $n \geq 1$ is the number of photos
    ▷ $E$ is a set of edges of the form $(p, p', s)$, where $s$ is the similarity of $p$ and $p'$
    ▷ $c$ is the number of categories, $1 \leq c \leq n$
    create a list of the edges of $E$, in increasing order by similarity
    let $\mathcal{C}$ be the categorization with all photos in one category
    Num-$\mathcal{C} \leftarrow 1$ ▷ initial number of categories
    while Num-$\mathcal{C} < c$ do
        remove the lowest-similarity edge $(p, p', s)$ from the list
        if $p$ and $p'$ are in the same category, say $S$, of $\mathcal{C}$ then
            ▷ split $S$ into two new categories $T$ and $T'$ as follows:
            put $p$ in $T$
            put $p'$ in $T'$
            for each remaining $p''$ in $S$ do
                let $s(p, p'')$ and $s(p', p'')$ be the similarities of $p$ and $p''$, and of $p'$ and $p''$, respectively
                put $p''$ in $T$ if $p''$ is more similar to $p$ than $p'$, i.e.,if $s(p, p'') > s(p', p')$
                put $p''$ in $T'$ otherwise
            ▷ now $S$ is replaced by $T$ and $T'$ in $\mathcal{C}$
            Num-$\mathcal{C} \leftarrow$ Num-$\mathcal{C} + 1$
    return $\mathcal{C}$
```

   What categorization does New-Clustering-Algorithm produce on the 4-node example given at the start of this question? Choose one.

   ○ $\{\{1,2\},\{3,4\}\}$ [✓ $\{\{1,3\},\{2,4\}\}$] ○ $\{\{1,4\},\{2,3\}\}$ ○ $\{\{1,2,3\},\{4\}\}$

3. Let (n, E, c) be an input where all edges have distinct similarities, and $c \geq 2$. Suppose that (p, p', s) is the edge removed in the *first* iteration of the while loop of New-Clustering-Algorithm. In an optimal Max-Min categorization, must p and p' be in distinct categories?

   [✓ Yes] ○ Not necessarily

4. Let E' be the set of all edges removed from the list, over all iterations of the while loop of New-Clustering-Algorithm, and let C^G be the categorization returned by the algorithm. Which statement is true? Choose one.

   [✓ All edges of E' must be inter-category edges of C^G]

   ○ All edges of E' must be intra-category edges of C^G

   ○ Edges of E' could be both intra-category and inter-category edges of C^G

5. Modify the 4-node example by choosing *one* edge from the list below, and then choosing a new similarity from the second list below for that edge, so that New-Clustering-Algorithm is *incorrect* on the resulting modified instance. That is, on the modified instance, New-Clustering-Algorithm produces a solution that does *not* maximize the minimum intra-cluster edge similarity.

   Choose one edge:

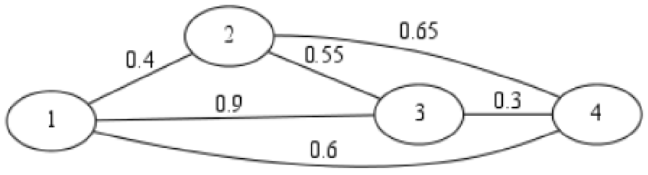
   Choose edge $\{2, 3\}$ and similarity .8. Then the first iteration creates categorization $\{\{1, 2, 3\}, \{4\}\}$, which is not optimal since category $\{1, 2, 3\}$ has an intra-category edge with cost .4.

   ○ $(1, 2)$ [✓ $(2, 3)$] ○ $(3, 4)$

   Choose the new edge similarity:

   ○ .5 [✓ .8]

The algorithm and example are shown again here for your reference.



```
function New-Clustering-Algorithm($n, E, c$)
    ▷ $n \geq 1$ is the number of photos
    ▷ $E$ is a set of edges of the form $(p, p', s)$, where $s$ is the similarity of $p$ and $p'$
    ▷ $c$ is the number of categories, $1 \leq c \leq n$
    create a list of the edges of $E$, in increasing order by similarity
    let $\mathcal{C}$ be the categorization with all photos in one category
    Num-$\mathcal{C} \leftarrow 1$ ▷ initial number of categories
    while Num-$\mathcal{C} < c$ do
        remove the lowest-similarity edge $(p, p', s)$ from the list
        if $p$ and $p'$ are in the same category, say $S$, of $\mathcal{C}$ then
            ▷ split $S$ into two new categories $T$ and $T'$ as follows:
            put $p$ in $T$
            put $p'$ in $T'$
            for each remaining $p''$ in $S$ do
                let $s(p, p'')$ and $s(p', p'')$ be the similarities of $p$ and $p''$, and of $p'$ and $p''$, respectively
                put $p''$ in $T$ if $p''$ is more similar to $p$ than $p'$, i.e., if $s(p, p'') > s(p', p')$
            put $p''$ in $T'$ otherwise
        $\triangleright$ now $S$ is replaced by $T$ and $T'$ in $\mathcal{C}$
        Num-$\mathcal{C} \leftarrow$ Num-$\mathcal{C} + 1$
**return** $\mathcal{C}$
```