1. (12 points) In the final analysis of the partition data structure, suppose that \( x \) is a non-singular node on level 2 and that \( \text{rank}(p(x)) = 11 \). What is the largest value that \( \text{rank}(x) \) can have? Explain.

*If \( x \) is on level 2, its rank is in a different level 1 block than the rank of its parent. So, \( \text{rank}(x) \) is at most 7.*

Now suppose that a *find* operation is executed and that \( x \) is one of the vertices on the “find path”. Give a lower bound on the value of \( \text{rank}(p(x)) \) after the find operation completes (your lower bound should be as large as possible and still be valid). Explain your answer.

*Since \( x \) is non-singular, it has an ancestor \( y \) on level 2 and \( y \) must have a parent whose rank is in the same level 2 block but a different level 1 block. Since the next block for both level 1 and level 2 starts at 16, \( y \) actually must have a rank of at least 16 and its parent must have a rank of at least 32. So the root of the tree must also have a rank of at least 32. Since the current root is \( x \)’s parent following the find, \( \text{rank}(p(x)) \) is at least 32 after the find.*

Is \( x \) still on level 2 after the find operation completes? Explain your answer.

*No, it is not, because \( \text{rank}(x) \) is still 7 which is block(2,1), while 32 is in block(2,2), so the rank of \( x \)’s parent is in a different level 2 block than \( \text{rank}(x) \).*
2. (12 points) This question concerns the round-robin algorithm for the minimum spanning tree problem. Assume that $G$ is a graph with 100 vertices, in which every vertex has 7 neighbors. Define a pass for the algorithm as follows; each pass ends when all the vertices that were on the queue at the start of the pass have been removed from the queue.

What is the smallest possible number of steps in the first pass (where a step is one iteration of the main loop)? What is the largest possible number of steps in the first pass?

The smallest possible number is 50. This occurs when the edges selected in the first pass form a matching.

The largest possible number is 99. This occurs when all steps after the first step connect the first vertex in the queue to one of the vertices that came before it in the queue, when the first pass started.

What is the smallest number of passes that can occur? What is the largest possible number of passes?

The smallest number is 1. The largest number is 6.

Suppose that at the start of the second pass, $u$ is the second vertex in the queue. What is the maximum number of dummy nodes that can be present in $h(u)$? If $h(u)$ has this maximum number of dummy nodes, how many vertices are in the queue at the start of the second pass?

$h(u)$ can have at most 97 dummy nodes, since the first item in the queue has at least 1, and at most 98 melds have occurred so far.

If $h(u)$ has 97 dummy nodes, then there are two vertices in the queue.

Suppose that at the start of the second pass, vertex $x$ is on the queue and that $h(x)$ contains three dummy nodes. What is the smallest possible number of nodes in $h(x)$ that correspond to deleted edges? What is the largest possible number?

If $h(x)$ has three dummy nodes, it was formed by combining four vertices into a single tree, using three edges. These edges correspond to 6 nodes in the heap, so there must be at least 6 nodes that correspond to deleted edges. These four vertices could have up to 6 edges joining them in the graph, so the number of nodes in the heap that correspond to deleted edges is at most 12.
3. (10 points) You are given a bipartite graph with edge weights and an integer $k$. The objective is to divide the graph into a minimum number of subgraphs, each of which can be edge-colored using $k$ colors. Assuming that the maximum vertex degree is $\Delta$, give an expression for the minimum number of subgraphs required (note: answer must be an integer).

*Since each subgraph can have vertex degree at most $k$, the minimum number of subgraphs is $\lceil \Delta/k \rceil$.*

For the graph shown below, show a solution to the problem for $k=2$, by writing a 1 next to all edges in the first subgraph, a 2 next to all edges in the second subgraph, etc.

![Graph](image)

Describe in words, an algorithm to solve this problem. You may use any algorithm we have discussed in class as a building block in your solution. However, your description of the solution must be precise, complete and unambiguous. Give an expression for the worst-case running time of your algorithm.

*We can apply the graph edge coloring algorithm from lab 3 to color the graph with $\Delta$ color. We can then use the first $k$ color sets to define the first subgraph, the next $k$ color sets to define the second subgraph, and so forth. The running time for this is $O(\Delta mn)$.*

*The problem can also be solved as a sequence of max flow problems. The algorithm is just like the flow-based algorithm for finding a maximum size matching in a bipartite graph. The only difference is that we let the capacity of the source/sink edges be $k$, not 1. After solving the first max flow problem, the edges in the central part with non-zero flow define a subgraph with maximum degree $k$. These are removed from the original graph before proceeding to the next stage. The next stage uses the same procedure as the first, defining a flow graph based on the “remainder subgraph”, finding a max flow in this subgraph, and so forth.*

*If we use the shortest augmenting path algorithm to find maximum flows, the running time is also $O(\Delta mn)$.***
4. (12 points) The diagram below shows a partial representation of an intermediate state in the execution of Edmonds’s algorithm for the maximum size matching problem.

Identify three blossoms in the graph by drawing closed curves around them. Recall that one blossom may be contained in another.

Write down the vertex sets defined by the partition data structure at this point in the execution of the algorithm. You need not show the structure of the trees.

\{a\}, \{b\}, \{u\}, \{v\}, \{g, m, n, s, t\}, \{c, d, e, h, i, j, k, o, p, x, w\}

Mark the bridges of the blossoms with a $B$.

If edge \{i, n\} is processed next, an augmenting path is found. List the vertices in that augmenting path.

\{a, b, h, c, d, j, e, k, x, w, p, o, i, n, t, s, m, g\}
5. (12 points) Consider the linear programming problem shown below in standard matrix form.

\[
\begin{array}{ccccc}
\text{maximize} & 1 & 2 & -1 & 1 \\
\text{subject to} & 2 & 0 & -1 & 1 \\
& 1 & 1 & 0 & 0 \\
& 0 & 2 & 0 & -1 \\
\end{array}
\begin{bmatrix}
x_a \\
x_b \\
x_c \\
x_d \\
\end{bmatrix}
\leq
\begin{bmatrix}
4 \\
3 \\
2 \\
\end{bmatrix}
\]

Write the dual for this linear program in matrix form.

\[
\begin{array}{cccc}
\text{minimize} & 4 & 3 & 2 \\
\text{subject to} & 2 & 1 & 0 & 1 \\
& 0 & 1 & 2 & 2 \\
& -1 & 0 & 0 & -1 \\
& 1 & 0 & -1 & 1 \\
\end{array}
\begin{bmatrix}
z_i \\
z_j \\
z_k \\
\end{bmatrix}
\geq
\begin{bmatrix}
1 \\
2 \\
-1 \\
\end{bmatrix}
\]

Suppose \([1 \ 1 \ 2 \ 1]^T\) is a feasible (but not necessarily optimal) solution to the primal problem and \([1 \ 3 \ 0]^T\) is a feasible solution to the dual problem. Write the slackness vector corresponding to the primal solution and the slackness vector corresponding to this dual solution.

*For the primal, the slackness vector is \([3 \ 1 \ 1]^T\). For the dual, the slackness vector is \([4 \ 1 \ 0 \ 0]^T\).*

Do these solutions satisfy the complementary slackness conditions? Based on this, what can you say about the optimality of the two given solutions?

*They do not satisfy the complementary slackness conditions, since the slackness vector \([3 \ 1 \ 1]\) and the vector of dual variables \([1 \ 3 \ 0]\) have non-zero values in some corresponding positions.*

*We can say that the given solutions are not both optimal (although, one could be).*

*We can also tell that the two solutions are not both optimal by comparing the objective function values for the given solutions (2 for the primal, 13 for the dual). Since these are not equal, the given solutions cannot both be optimal. Consequently, they cannot satisfy the complementary slackness conditions.*
6. (12 points). The diagram below shows an intermediate state in the augmenting path algorithm for the weighted matching problem in bipartite graphs. The numbers next to the vertices are the labels.

List all of the equality edges in the graph.

Edges cy, cz, by, bu, dx and dv are all equality edges.

Mark up the diagram to show the state of the algorithm after it has built the largest forest that it can without changing the vertex labels. Mark even vertices with a +, odd vertices with a −, and use arrows to indicate the parent pointers in the collection of trees.

What is the value of $\delta$ in the next relabeling step? After the relabeling step, what are the labels for vertices $a$, $b$, $c$, $d$ and $e$?

$\delta=0.5$

$z_a=2.5$, $z_b=5.5$, $z_c=3.5$, $z_d=3.5$ and $z_e=2.5$
7. (15 points) This question concerns Edmond’s algorithm for finding weighted matchings in general graphs. Suppose G is a graph in which the edge weights range from 5 to 50. Suppose that the first four label adjustments have δ=2, 6, 7 and 5. Suppose vertex u is not matched after these first four label adjustments have been made. What is the value of z_u?

The initial label of u was 25 and since u is still free, its label was reduced by δ during each of the label adjustments, so z_u=25−(2+6+7+5)=5.

Suppose B_1 is a blossom consisting of edges {a, b}, {b, c} and {c, a} that is contained within another blossom B_2 with edges {b, d}, {d, e} and {c, e}. Assume that B_2 is odd and that there are no other blossoms. Let {a, x} be a matching edge and let {e, y} and {y, v} be non-matching edges with w(e, y)=4, w(v, y)=5, y unreached and v even. The table below lists the labels for these vertices and blossoms.

<table>
<thead>
<tr>
<th>u</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>v</th>
<th>x</th>
<th>y</th>
<th>B_1</th>
<th>B_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>z_u</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

What are the values of the edge weights w(a, b), w(d, e) and w(a, x)?

These must all be equality edges. Since {a, x} is not in any blossom, its edge weight is just the sum of its endpoints’ labels, so w(a, x)=10. {d, e} is contained in B_2 but not B_1, so w(d, e)=z_d+z_e+z_{B_2}=12. {a, b} is contained in both blossoms, so w(a, b)=z_a+z_b+z_{B_1}+z_{B_2}=13.

Based on the information given above, if the next step adjusts the label values, what is the value of δ that is used?

δ_1=4, δ_2=4, δ_3 is undefined and δ_4=3, so δ=1.5.

After the label adjustment, what are the values of z_a, z_d, z_e and z_y?

z_a=6.5, z_d=4.5, z_e=3.5, z_y=6