Question 1A Collision [1 marks]

◯ True
◯ False

False

Question 1B Expectations [1 marks]

◯ True
◯ False

True

Question 1C Expectations again [1 marks]

◯ True
◯ False

False

Question 1D Implementation of hashCode() [1 marks]

◯ True
◯ False

False

Question 1E Heapsort [1 marks]

◯ True
◯ False

False

Question 1F Size of heaps [1 marks]

◯ True
◯ False

True

Question 1G Cost of merging heaps [1 marks]

◯ True
◯ False

True

Question 1H Open addressing vs Chaining [1 marks]

○ True

○ False

True

Question 1I Amortized analysis [1 marks]

○ True

○ False

False

Question 1J DFS in adjacency matrix model [1 marks]

○ True

○ False

True

Question 1K Adding 5 [1 marks]

○ True

○ False

False

Question 1L BFS on weighted tree [1 marks]

○ True

○ False

True

Question 1M DFS on weighted tree [1 marks]

○ True

○ False

True

Question 1N BFS layers and path lengths [1 marks]

○ True

○ False

False

Question 1O Dijkstra relaxations [1 marks]

- ◯ True
- ◯ False

False

Question 1P Recurrence [1 marks]

- ◯ True
- ◯ False

False

Question 1Q Best-case running time for quicksort [1 marks]

- ◯ True
- ◯ False

True

Question 1R Insertions in AVL [1 marks]

- ◯ True
- ◯ False

False

Question 1S How many rotations to balance an unbalanced tree [1 marks]

- ◯ True
- ◯ False

False

Question 2A Expected number of length-1 chains [4 marks]

- ◯ n/m
- ◯ $1+n/m$
- ◯ $(1-1/m)^{n-1}$
- ◯ $(1-1/m)^{m-1}$
- ◯ $n(1-1/m)^{n-1}$
- ◯ $n(1-1/m)^{m-1}$

$n(1-1/m)^{n-1}$

Question 2B Probability of no collision [3 marks]

- ○ k/m
- ○ $1-k/m$
- ○ $(1-k/m)^m$
- ○ $(k/m)^m$
- ○ $(1-k/m)^{m-1}$
- ○ $(k/m)^{m-1}$

$1-k/m$

Question 2C Probability of < 2 probes [4 marks]

- ○ k/m
- ○ $1-k/m$
- ○ k^2/m^2
- ○ $1-k^2/m^2$
- ○ $(k^2-k)/(m^2-m)$
- ○ $1-(k^2-k)/(m^2-m)$

$1-k/m$

Question 2D Expected number of insertions needing 1 probe [4 marks]

- ○ $3n/4$
- ○ $(3n+1)/4$
- ○ $n/2$
- ○ $(n+1)/2$
- ○ $n/4$
- ○ $(n+3)/4$

$(3n+1)/4$

Question 2E Probability of one probe with length-c chains [3 marks]

- ○ m_c/m
- ○ $1-m_c/m$
- ○ $1-m_cm_{c-1}/m^2$
- ○ m_cm_{c-1}/m^2
- ○ m_0/m
- ○ $1-m_0/m$

$1-m_c/m$

Question 2F Probability of $m_2 = m_3 = \cdots = m_c = 0$ after n insertions [4 marks]

- ○ $\frac{m!}{m^n(m-n)!}$
- ○ $\frac{m!}{m^c(m-n)!}$
- ○ $(1-\frac{1}{m})^n$
- ○ $(1-\frac{1}{m})^c$
- ○ $1-\frac{n^2}{m}$
- ○ $1-\frac{nc}{m}$

$\frac{m!}{m^n(m-n)!}$

Question 3A Is the graph a DAG? [1 marks]

- ○ True
- ○ False

False

Question 3B Pre-order DFS traversal [2 marks]

- ○ s,a,b,c,d,e,f
- ○ s,a,b,d,c,e,f
- ○ s,b,e,a,c,f,d
- ○ s,b,e,a,d,f,c
- ○ c,f,e,b,d,a,s
- ○ c,a,d,f,b,e,s

s,b,e,a,d,f,c

Question 3C Post-order DFS traversal [2 marks]

- ○ s,a,b,c,d,e,f
- ○ s,a,b,d,c,e,f
- ○ s,b,e,a,c,f,d
- ○ s,b,e,a,d,f,c
- ○ c,f,e,b,d,a,s
- ○ c,a,d,f,b,e,s

c,f,e,b,d,a,s

Question 3D How to avoid dragons on edges? [2 marks]

- ◯ Breadth-first search
- ◯ Depth-first search
- ◯ Bellman-Ford
- ◯ Dijkstra
- ◯ Relax in topological order

Dijkstra

Question 3E How to avoid dragons on nodes and edges? [4 marks]

- ◯ Each existing edge e has weight n_e. At each node v, create a new self-loop edge (v, v) with weight n_v.
- ◯ For each edge e connecting node u to node v, let it have weight $n_e + n_u$.
- ◯ For each edge e connecting node u to node v, let it have weight $n_e + n_v$.
- ◯ None of the above.

For each edge e connecting node u to node v, let it have weight $n_e + n_v$.

Question 3F How to avoid spells? [4 marks]

◯ Perform a BFS traversal from s, and check if the path from s to t (induced by the parent pointers) contains at most one node in X.

◯ Perform a DFS traversal from s, and check if the path from s to t (induced by the parent pointers) contains at most one node in X.

◯ Create a new graph G'. For each town v, create two nodes v_0 and v_1 in G'. For each road (u,v), create three edges (u_0,v_0), (u_0,v_1) and (u_1,v_1). Check (using BFS or DFS) whether t_1 is reachable from s_0 in G'.

◯ Create a new graph G'. For each town v, create two nodes v_0 and v_1 in G'. For each road (u,v), if $u \in X$, create an edge (u_0,v_1), and if $u \notin X$, create two edges (u_0,v_0) and (u_1,v_1). Check (using BFS or DFS) whether t_1 is reachable from s_0 in G'.

◯ Create a new graph G'. For each town v, create two nodes v_0 and v_1 in G'. For each road (u,v), if $u \in X$, create an edge (u_0,v_1), and if $u \notin X$, create two edges (u_0,v_0) and (u_1,v_1). Check (using BFS or DFS) whether either t_0 or t_1 is reachable from s_0 in G'.

◯ Create a new graph G'. For each town v, create two nodes v_0 and v_1 in G'. For each road (u,v), if $v \in X$, create an edge (u_0,v_1), and if $v \notin X$, create two edges (u_0,v_0) and (u_1,v_1). Check (using BFS or DFS) whether t_1 is reachable from s_0 in G'.

◯ Create a new graph G'. For each town v, create two nodes v_0 and v_1 in G'. For each road (u,v), if $v \in X$, create an edge (u_0,v_1), and if $v \notin X$, create two edges (u_0,v_0) and (u_1,v_1). Check (using BFS or DFS) whether either t_0 or t_1 is reachable from s_0 in G'.

◯ None of the above.

Create a new graph G'. For each town v, create two nodes v_0 and v_1 in G'. For each road (u,v), if $v \in X$, create an edge (u_0,v_1), and if $v \notin X$, create two edges (u_0,v_0) and (u_1,v_1). Check (using BFS or DFS) whether either t_0 or t_1 is reachable from s_0 in G'.

Question 3G How to minimize energy spent on spells? [7 marks]

Solution Sketch: Create a new graph G'. For each town v, create nodes $v_0, v_1, \ldots, v_{Z_v}$ in G', and add edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{Z_v - 1}, v_{Z_v})$, each with weight 1, to form a path from v_0 to v_{Z_v} in G'. For any road from u to v, create an edge u_{Z_u} to v_0 in G' with weight 0.

The total number of nodes in G' is $\sum_v (Z_v + 1) = Z + n = O(Z)$, since $Z \geq m \geq n - 1$ where the first inequality is by the problem's assumption on Z and the second inequality is because the original graph is connected. The total number of edges is $\sum_v Z_v + m = O(Z)$, because $Z \geq m$ by assumption.

Now, we can run an SSSP algorithm from s, where all the edge weights are either 0 or 1. We can use Dijkstra and apply the standard analysis, but this would have running time $O(Z \log Z)$. We can get an $O(Z)$ time algorithm by noticing that there are $O(Z)$ possibilities for distances from s, since there are $O(Z)$ many nodes and each edge has weight either 0 or 1. So, instead of using a general priority queue, we can keep $O(Z)$ many queues for vertices at each distance, giving constant time EXTRACT-MIN and DECREASE-KEY implementations. (In fact, only at most two queues will be non-empty at any point in Dijkstra. Prove this!)

Grading scheme: 3 points for correctly creating an edge-weighted graph. People did this in mostly two ways. One option was as above. The other option was to have edge (u, v) have weight Z_v.

2 points for applying the standard version of Dijkstra with $O(Z \log Z)$ running time. 4 points for modifying Dijkstra to get $O(Z)$ runtime.

A common approach was running BFS instead of Dijkstra. This doesn't work because edges have unequal weights. Some people got around this by replacing each edge (u, v) with a path of length Z_v, and then running BFS. But this can make the number of new nodes be $O(Z^2)$ (imagine a star where the hub has degree m and also energy cost m). BFS without correct analysis got 0 points.

Some people also claimed $O(Z)$ runtime by relaxing edges in topological order. However, the graph was not given to be a DAG.

The most common (nonzero) score was 5: 3 points for correct edge weights and 2 points for mentioning Dijkstra and correct runtime analysis.

Question 4A Fill in A [2 marks]

○ 0	○ `begin/2`	○ `>=`	○ `mid+1`
○ 1	○ `<`	○ `=`	
○ −1	○ `>`	○ `mid-1`	
○ `begin`	○ `<=`	○ `mid`	

1

Question 4B Fill in B [2 marks]

○ 0	○ `begin/2`	○ `>=`	○ `mid+1`
○ 1	○ `<`	○ `=`	
○ −1	○ `>`	○ `mid-1`	
○ `begin`	○ `<=`	○ `mid`	

>=

Question 4C Fill in C [2 marks]

○ 0	○ `begin/2`	○ `>=`	○ `mid+1`
○ 1	○ `<`	○ `=`	
○ −1	○ `>`	○ `mid-1`	
○ `begin`	○ `<=`	○ `mid`	

`mid`

Question 4D Fill in D [2 marks]

○ 0	○ `begin/2`	○ `>=`	○ `mid+1`
○ 1	○ `<`	○ `=`	
○ −1	○ `>`	○ `mid-1`	
○ `begin`	○ `<=`	○ `mid`	

`mid-1`

Question 5A Dijkstra's implementation of Dijkstra [3 marks]

- ◯ Dijkstra's implementation has better asymptotic running time for sparse graphs (small E)
- ◯ Dijkstra's implementation has better asymptotic running time for dense graphs (large E)
- ◯ Dijkstra's implementation always has better asymptotic running time but has higher memory usage.
- ◯ The standard implementation always has better asymptotic running time than Dijkstra's version.

Dijkstra's implementation has better asymptotic running time for dense graphs (large E)

Question 5B Minimum-weight path with at least 20 hops [4 marks]

◯ Run BFS on G from s. Find a vertex v which is on level 20 of the BFS tree; let P_1 be the BFS tree path from s to v. Run Bellman-Ford on G from v, and find a minimum weight path P_2 from v to t. Output the concatenation of P_1 and P_2.

◯ Run Bellman-Ford on G from s. Find a vertex v which is on level 20 of the shortest-path tree; let P_1 be the minimum weight path from s to v. Run Bellman-Ford on G from v, and find a minimum weight path P_2 from v to t. Output the concatenation of P_1 and P_2.

◯ Let G_2 be G but with edge orientations reversed. Run BFS on G_2 from t. Find a vertex v which is on level 20 of the BFS tree; let P_1 be the BFS tree path from t to v. Run Bellman-Ford on G_2 from v, and find a minimum weight path P_2 from v to s. Concatenate P_1 and P_2, reverse the edge orientations, and output.

◯ Let G_3 be the weighted graph, which for each vertex v in G, has 21 vertices $v_0, v_1, \ldots, v_{20}$, and for each edge (u,v) in G, has 20 edges $(u_0,v_1),(u_1,v_2),\ldots,(u_{19},v_{20})$, each with the same weight as (u,v) in G. Run Bellman-Ford on G_3 from s_0, and let w in G be such that the path from s_0 to w_{20} has the smallest weight in G_3. Let P_1 be the corresponding path of 20 edges from s to w in G. Run Bellman-Ford on G from w, and let P_2 be a minimum weight path from w to t. Output the concatenation of P_1 and P_2.

◯ Let G_3 be as in the 4th option above. Run Bellman-Ford on G_3 from s_0. For each vertex v in G, let $\delta_1(v)$ be the minimum weight of a path from s_0 to v_{20} in G_3 and $P_{1,v}$ the corresponding path of 20 edges from s to v in G. Let G_2 be as in the 3rd option above. Run Bellman-Ford on G_2 from t. For each vertex v in G, let $\delta_2(v)$ be the minimum weight of a path from t to v in G_2 and let $P_{2,v}$ the corresponding path in G from v to t. Let w be the vertex that minimizes $\delta_1(w)+\delta_2(w)$; output the concatenation of $P_{1,w}$ and $P_{2,w}$.

◯ None of the above.

Let G_3 be as in the 4th option above. Run Bellman-Ford on G_3 from s_0. For each vertex v in G, let $\delta_1(v)$ be the minimum weight of a path from s_0 to v_{20} in G_3 and $P_{1,v}$ the corresponding path of 20 edges from s to v in G. Let G_2 be as in the 3rd option above. Run Bellman-Ford on G_2 from t. For each vertex v in G, let $\delta_2(v)$ be the minimum weight of a path from t to v in G_2 and let $P_{2,v}$ the corresponding path in G from v to t. Let w be the vertex that minimizes $\delta_1(w)+\delta_2(w)$; output the concatenation of $P_{1,w}$ and $P_{2,w}$.

Question 5C Weight transformation preserving shortest paths [3 marks]

- ◯ In G', the weight of an edge e is $w'_e = w_e + 1$.
- ◯ In G', the weight of an edge e is $w'_e = w_e^2$.
- ◯ In G', the weight of an edge $e = (u, v)$ is $w'_e = 2w_u + w_e$.
- ◯ In G', the weight of an edge $e = (u, v)$ is $w'_e = 2w_u - 2w_v + w_e$.
- ◯ In G', the weight of an edge $e = (u, v)$ is $w'_e = 2w_u - 3w_v + w_e$.
- ◯ In G', the weight of an edge $e = (u, v)$ is $w'_e = 2w_u - 3w_v$.
- ◯ In G', the weight of an edge $e = (u, v)$ is $w'_e = 2w_u w_e$.

In G', the weight of an edge $e = (u, v)$ is $w'_e = 2w_u - 2w_v + w_e$.

Question 5D Weight transformation preserving MST [3 marks]

- ◯ In G', the weight of an edge e is $w'_e = w_e + 1$.
- ◯ In G', the weight of an edge e is $w'_e = w_e^2$.
- ◯ In G', the weight of an edge $e = \{u, v\}$ is $w'_e = 2w_u + 2w_v + w_e$.
- ◯ In G', the weight of an edge $e = \{u, v\}$ is $w'_e = 2w_u^2 + 2w_v^2 + w_e$.
- ◯ In G', the weight of an edge $e = \{u, v\}$ is $w'_e = 2w_u + 2w_v$.
- ◯ In G', the weight of an edge $e = \{u, v\}$ is $w'_e = 2w_u w_v w_e$.

In G', the weight of an edge e is $w'_e = w_e + 1$.

Question 5E Updating MST after weight decrease [3 marks]

- (I) ◯ The proposed algorithm is correct.
- (II) ◯ The proposed algorithm runs in time $O(V + E)$.
- (III) ◯ Both (I) and (II) are correct.
- (IV) ◯ Neither (I) and (II) is correct.

Both (I) and (II) are correct.

Question 6A Base case for stacking problem [2 marks]

- ◯ $T[0, h_1, h_2, h_3]$ = True if $h_1 = h_2 = h_3 = 0$, and False otherwise.
- ◯ $T[0, h_1, h_2, h_3]$ = True if $h_1 = 0$ or $h_2 = 0$ or $h_3 = 0$, and False otherwise.
- ◯ $T[1, h_1, h_2, h_3]$ = True if $h_1 = h_2 = h_3 = 0$, and False otherwise.
- ◯ $T[1, h_1, h_2, h_3]$ = True if $h_1 = 0$ or $h_2 = 0$ or $h_3 = 0$, and False otherwise.
- ◯ None of the above.

$T[0, h_1, h_2, h_3]$ = True if $h_1 = h_2 = h_3 = 0$, and False otherwise.

Question 6B Recurrence for stacking problem [4 marks]

- ◯ $T[i,h_1,h_2,h_3] = T[i-1,h_1-z_i,h_2-z_i,h_3-z_i]$
- ◯ $T[i,h_1,h_2,h_3] = (h_1 \geq z_i$ OR $h_2 \geq z_i$ OR $h_3 \geq z_i)$ AND $T[i-1,h_1-z_i,h_2-z_i,h_3-z_i]$
- ◯ $T[i,h_1,h_2,h_3] = (h_1 \geq z_i$ OR $h_2 \geq z_i$ OR $h_3 \geq z_i)$ AND $T[i-1,h_1-z_i,h_2-z_i,h_3-z_i]$
- ◯ $T[i,h_1,h_2,h_3] = T[i-1,h_1-z_i,h_2,h_3]$ OR $T[i-1,h_1,h_2-z_i,h_3]$ OR $T[i-1,h_1,h_2,h_3-z_i]$
- ◯ $T[i,h_1,h_2,h_3] = (h_1 \geq z_i$ AND $T[i-1,h_1-z_i,h_2,h_3])$ OR $(h_2 \geq z_i$ AND $T[i-1,h_1,h_2-z_i,h_3])$ OR $(h_3 \geq z_i$ AND $T[i-1,h_1,h_2,h_3-z_i])$
- ◯ $T[i,h_1,h_2,h_3] = (h_1 \geq z_i$ AND $T[i-1,h_1-z_i,h_2,h_3])$ AND $(h_2 \geq z_i$ AND $T[i-1,h_1,h_2-z_i,h_3])$ AND $(h_3 \geq z_i$ AND $T[i-1,h_1,h_2,h_3-z_i])$

$T[i,h_1,h_2,h_3] = (h_1 \geq z_i$ AND $T[i-1,h_1-z_i,h_2,h_3])$ OR $(h_2 \geq z_i$ AND $T[i-1,h_1,h_2-z_i,h_3])$ OR $(h_3 \geq z_i$ AND $T[i-1,h_1,h_2,h_3-z_i])$

Question 6C Running time for stacking problem [3 marks]

- ◯ $O(n^4)$
- ◯ $O(nZ^3)$
- ◯ $O(n^2Z^2)$
- ◯ $O(nZ^2)$
- ◯ $O(Z^4)$
- ◯ None of the above

$O(nZ^2)$. This was a bit tricky. Note that for any fixed i, the algorithm evaluates $T[i,h_1,h_2,h_3]$ for $\leq Z^2$ values of (h_1,h_2,h_3), not Z^3, since $h_1+h_2+h_3$ equals $\sum_{j=1}^{i} z_j$ on any recursive call.

Question 6D Counting number of subsequence occurrences [4 marks]

◯ Maintain $A[i][j]$ to be the count of the occurrences of $Y[1\ldots i]$ as a subsequence in $X[1\ldots j]$. Use base case $A[0][0] = A[0][1] = \cdots = A[0][n] = 0$, and apply recurrence $A[i][j] = A[i-1][j-1]+1$.

◯ Maintain $B[i][j]$ to be the count of the occurrences of $Y[1\ldots i]$ as a subsequence in $X[1\ldots j]$. Use base case $B[0][0] = B[1][0] = \cdots = B[m][0] = 0$, and apply recurrence: $B[i][j] = B[i][j-1]$ if $Y[i] \neq X[j]$ and $B[i][j] = B[i][j-1] + B[i-1][j-1]$ if $Y[i] = X[j]$.

◯ Maintain $C[i][j]$ to be the count of the occurrences of $Y[1\ldots i]$ as a subsequence in $X[1\ldots j]$. Use base case $C[0][0] = C[0][1] = \cdots = C[0][n] = 0$, and apply recurrence: $C[i][j] = C[i-1][j]$ if $Y[i] \neq X[j]$ and $C[i][j] = C[i-1][j] + C[i-1][j-1]$ if $Y[i] = X[j]$.

◯ Maintain $D[i][j]$ to be the count of the occurrences of $Y[i\ldots j]$ as a subsequence in X. Let n_i be number of occurrences of $Y[i]$ in X. Use base case: for every i, $D[i][i] = n_i$. Apply recurrence: $D[i][j] = D[i+1][j-1] \cdot n_i \cdot n_j$.

◯ Maintain $E[i][j]$ to be the count of the occurrences of Y as a subsequence in $X[i\ldots j]$. Use base case $E[0][0] = 0$. Apply recurrence: $E[i][j] = E[i][\lfloor (i+j)/2 \rfloor] + E[\lfloor (i+j)/2 \rfloor + 1][j]$.

◯ None of the above.

None of the above.