CIT 5960 Online

Midterm 2 Solutions

1. (a) False. Consider a graph where the vertex n has a single edge incident on it that has weight m. The remaining $(n-1)$ vertices form a connected graph with$(m-1)$ edges that have distinct weights $1, 2, ..., (m-1)$. Since $(m-1) \geq (n-2)$, this is always possible. Now, any MST must contain the unique edge incident on vertex n.

   (b) True. In the greedy merging sequence, since we always choose to merge two symbols of smallest probability, symbol 1 will be merged for the first time only when there are 2 symbols remaining. The process terminates at this point, and hence symbol 1 is assigned a code of length 1.

2. (a) *Algorithm* Use BFS/DFS to find the number of connected components in G. If this number is K or larger, we can assign a distinct color to the vertices in each connected component. On the other hand, if this number is less than K, we output that there is no consistent coloring that uses
K or more co

   Justification. of G must get the same
color in any nnected component are
connected by of G can not use more
colors than t

   Runtime. As complexity.

   (b) *Algorithm* C . If G is strongly con-
nected, then $, E_{SCC})$ be the acyclic
component g $, c_k$ be the sequence of
topologically onnected component in
G correspond in C_k to any vertex in
C_1. Test if th output e as the desired
edge, else ou

   Justification. onnected, we claim that
it must be t here are no edges going
out of comp $G' = G + e$ is strongly
connected, th C_i. Moreover, if *some*
edge e from C_k to C_1 must make G
strongly-conn connected components.
Thus, G is al add makes G' strongly
connected.

   Runtime. Computing the strongly connected component graph and topologically sorting it take $O(|V| + |E|)$ time each. Adding the edge e takes constant time, and testing if G' is strongly connected also takes $O(|V| + |E|)$, so the entire algorithm takes $O(|V| + |E|)$ time.

   (c) *Algorithm:* Let the edge $e = (u, v)$. Let G' be the graph obtained by deleting edge e from the graph. Run Dijkstra's algorithm on G' to find the shortest path from v to u. If Dijkstra's algorithm outputs "no feasible path", then output "none". Else, if T is the cost of the shortest path, then output $T + \mathsf{cost}(e)$ where $\mathsf{cost}(e)$ is the weight of the edge e.

   Justification: Consider any cycle in G in which (u, v) is an edge. Tracing the edges of the cycle starting from v, we see that such a cycle yields a simple path from v to u (which of course does not include the edge (u, v)). In the other direction, given any simple path from v to u, we can add the edge (u, v) to get a cycle. Thus, the cycles containing (u, v) are in one-to-one correspondence with simple paths from v to u. Further, the total weight of the cycle is exactly the same as the weight of the corresponding path + the weight of the edge (u, v). This finishes the proof.

   Running Time: Given the adjacency list representation of G, the adjacency list representation of G' can be computed in time $O(m + n)$. Subsequently, the cost of running Dijkstra is $O(m \log n)$ which gives us total bound on the running time.

3. **Algorithm:** Let $H = \sum_{i=1}^{n} h_i$ denote the total happiness associated with all n toys; note that H is an integer between 0 and nM. We now define our subproblems. For each integer $0 \leq i \leq n$ and $0 \leq j \leq nM$, let $T[i,j]$ be 1 if there is a subset of the first i items whose total happiness is exactly j, and 0 otherwise. The base case of this computation is as below:

$$T[0,j] = \begin{cases} 1 & \text{if } j = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Now for the inductive computation, we define as follows:

$$T[i,j] = \begin{cases} 1 & \text{if } T[i-1,j] = 1 \\ 1 & \text{if } j \geq h_i \text{ and } T[i-1,j-h_i] = 1 \\ 0 & \text{otherwise.} \end{cases}$$

We will compute the entries in the table T, in increasing order of i, and for each i, we compute it in increasing order o [illegible] manner, all subproblems on which it depen [illegible]

Finally, we output [illegible] else not.

Proof of correc[illegible] ase case corresponds to $i = 0$, and $T[0,j]$ [illegible] ting an empty set. The only achievable ha [illegible]

Now, assume by [illegible] all $i' < i$, and for all $0 \leq j \leq nM$. T [illegible] the first i items whose happiness is exact [illegible] oy i or it does. If such a subset does not in [illegible] r hand, if such a subset includes toy i, it n [illegible] $-1)$ items that gives the remaining happine [illegible] 1. If neither of the two cases occur, then [illegible] and we can correctly set $T[i,j] = 0$.

Finally, note that [illegible] n toys whose happiness is exactly $H/2$.

Time complexit [illegible] $[0,j]$ for all $0 \leq j \leq nM$ can be done in $O($ [illegible] $i' < i$, the entry $T[i,j]$ can be computed [illegible] table T is $O(n^2 \cdot M)$. Finally, the last st [illegible] e total time complexity is $O(n^2 M)$.