Question 3

Solution. Consider the following algorithm solving GOLDDIGGEROPT, which uses the algorithm for the GOLDDIGGER decision problem. Note that the notation $A[i:]$ is used to denote a matrix consisting of the rows of A from the i^th row downwards (including the i^th row). An explanation of the algorithm follows.

```
GOLDDIGGEROPT(H, G, h):
 1  g ← greatest value that satisfies GOLDDIGGER(H, G, h, g).
 2  A ← n + 1 by m matrix where the first row is filled with (0, 0) and the rest is filled with NIL.
 3  // A is used to keep track of promising subpaths, where each entry is a pair representing the
 4  // h and g up to that point.
 5  for i = 1 to n:
 6      for k = 0 to m − 1:
 7          // if on a promising subpath
 8          if A_{i−1,k} ≠ NIL:
 9              // check the possible extensions of the subpath
10              for j = k − 1 to k + 1, 0 ≤ j ≤ m − 1:
11                  if GOLDDIGGER(H[i − 1 :], G[i − 1 :], h − A_{i−1,j}[0], g − A_{i−1,j}[1]) = True:
12                      A_{i,j} ← (A_{i−1,j}[0] + H_{i−1,j}, A_{i−1,j}[1] + G_{i−1,j})
13                  if A_{i,j}[1] = g:
14                      return A, i, j
```

Given A, i, and j, we can reconstruct the solution in a way similar the algorithm in A1, so the pseudocode for the algorithm is omitted here.

First, we find the most amount of gold attainable with the given H, G, and h on line 1. Using binary search with $g_l = 0$ as the lower bound and $g_u = \Sigma_{i=1}^{n}\Sigma_{j=1}^{m} G_{i,j}$ as the upper bound to find the first guess (if GOLDDIGGER$(H, G, h, guess)$ is False, $guess$ becomes the new upper bound and if it is True, then $guess$ becomes the new lower bound), even if the size of the upper bound is exponential, running binary search on it would take polynomial time. Now given this optimal value for g, we look for the path which leads to it using the GOLDDIGGER decision problem algorithm.

First, we look at the top row of H and G. For each element in this row, say $H_{1,j}$ and $G_{1,j}$, we check whether it is not on the optimal path by running GOLDDIGGER$(H[1:], G[1:], h - H_{1,j}, g - G_{1,j})$. If this returns false, choosing to dig at the block $(1, j)$ will make it impossible to obtain the optimal amount of gold. Hence, we conclude that it cannot be on the optimal path. If the decision problem algorithm returns True, then the block might be on the optimal path. By querying each block of the first row in turn, we eliminate any blocks that cannot be in the final solution and keep any blocks that are promising. Then, we repeat the same process of using the decision problem to eliminate non-promising subpaths on all of the blocks reachable from the promising blocks from the previous iteration until we find a path with the optimal g and a valid h.

Since finding the optimal g takes polynomial time, and GOLDDIGGER is called at most once for each block in H and G, if GOLDDIGGER runs in polynomial time then GOLDDIGGEROPT runs in $O(P_1 + m * n * P_2)$, where P_1 is the time it takes to find the optimal g and P_2 is the time it takes to run GOLDDIGER. Therefore, if GOLDDIGGER takes polynomial time, then GOLDDIGGEROPT does as well.

By the correctness of the GOLDDIGGER decision problem algorithm, we know that there exists a path with hardness $\leq h$ and gold $= g$, and that for any valid path with hardness h_0 and gold g_0, if this path is a subpath of the optimal path, then GOLDDIGGER$(H, G, H - h_0, G - g_0)$ must return true. Hence, at any point in the algorithm, one of the promising subpaths must be a subpath of the optimal path, and eventually the algorithm will find the optimal path for the given g. Since by definition g is the highest possible gold attainable with the given h, this algorithm solves the GOLDDIGGEROPT problem. ♦