
Problem 1

Initial Configuration

Let us consider the first row. We must ensure that $f(1,1) = 1$. We know that the first input symbol of the input is at position $(1,1)$ on the tape, so we can label the tile at $(1,1)$ with $(\langle w_1, q_s\rangle, \langle w_2, w_3\rangle)$. The label identifies the first symbol of the input tape, w_1, which is blank; the current state of M, q_s; and the two neighboring symbols to the right of the blank symbol, w_2 and w_3, which are also blank. Since the only symbol that fits in $(1,1)$ is blank, we can verify that $f(1,1) = 1$. Next, we need markers that identify the end of a row, so that we can transition to the next row. Using the hint, we can use triplets of symbols to name the tiles, specifically by their neighbors. Doing this will allow us to identify the valid configurations of subsequent triplets given our known ones. Additionally, we can use the tuple (a, q) where a is a symbol read from the tape and q is a state in the Turing Machine to represent a tile (TA help).

TILE instance

We can now construct a tiling instance $\langle k, H, V\rangle$ from a given instance $\langle M, w\rangle$ of $\overline{\text{HALT}}$. Let $k = 3$, and we will define the tile set to consist of tiles of the form:

$$[(a,q),(b,r),(c,s)] \quad , \quad [(p_1,q_1),(p_2,q_2),(p_3,q_3)]$$

where a, b, c, p_1, p_2, p_3 are tape symbols and q, r, s, q_1, q_2, q_3 are states, and the following conditions hold:

- The triplets $(a, p_1, p_2), (b, p_2, p_3), (c, p_3, q)$ are legal transitions of M. That is, there exist transitions of the form (q, a, b, R, r), (r, p_1, p_2, R, s), (s, p_2, p_3, R, q') in the transition function of M, where q' is either q, r, s, or a halting state.
- The states q_1, q_2, q_3 are such that $(a, q_1), (b, q_2), (c, q_3)$ are the symbols that appear to the left of the triplets $(a, p_1, p_2), (b, p_2, p_3), (c, p_3, q)$ in the previous row of the tiling.
- The pairs $(p_1, q_1), (p_2, q_2), (p_3, q_3)$ are such that (p_1, q_1) is a legal transition from the symbol to its left, and $(p_2, q_2), (p_3, q_3)$ are legal transitions from the symbols to their left and their right, respectively, according to the transition function of M.

Horizontal and Vertical Compatibility

We will define the horizontal and vertical compatibility relations H and V as follows:

- For all tiles t_1, t_2 such that t_1 appears to the right of t_2, we add (t_1, t_2) to H if the middle symbol in t_1 is the same as the left symbol in t_2.
- For all tiles t_1, t_2 such that t_1 appears below t_2, we add (t_1, t_2) to V if the triplets of symbols in t_1 and t_2 are compatible according to the rules above.

The first condition on the tiles ensures that a legal sequence of transitions in M corresponds to a legal sequence of tiles in the tiling. In particular, the triplets of symbols in each tile correspond to the symbols that M writes to the tape in a single step, so the vertical compatibility condition ensures that these symbols are consistent with the rules. We can ensure that any two tiles representing symbols that are adjacent in a configuration must be vertically compatible. Similarly, any two tiles representing the same symbol must be horizontally compatible.

Diagonal Compatibility

Now, we also add some extra constraints to ensure that any two tiles representing triplets of symbols that are adjacent in a configuration must be diagonally compatible. For each configuration C of the modified Turing machine M on input w, we define a tile T_C with the following properties:

- The first row of tiles encodes the initial configuration of M on input w, with the first tile being $T_{C_{\text{start}}}$, where C_{start} is the initial configuration of M on input w.
- For each subsequent row i, we define a set of tiles T_{C_j}, such that any valid tiling of the i^{th} row can be extended to a valid tiling of the $(i+1)^{th}$ row by adding tiles from this set. This set consists of all tiles T_C such that C is a legal configuration of M that can be reached from the configuration corresponding to the tiles in the previous row.

We add extra constraints on the compatibility of the tiles to ensure that any two tiles representing triplets of symbols that are adjacent in a configuration are diagonally compatible. Specifically, we add a diagonal compatibility relation between any two tiles T_C and $T_{C'}$ if there exist two adjacent symbols in C and C' such that the corresponding tiles are horizontally or vertically adjacent in the tiling. For any state q and symbol a:

- If $(q, a) \notin T$, then M' halts and rejects the input. If $(q, a) = (q', b, d)$, then M' writes the symbol (b, q') on the tape at the current position, moves the head according to d, and enters state q'. For any state q and symbol (a, r) where r is neither accept or reject,
- If $(r, a, d) \notin T$, then M' halts and rejects the input. If $(r, a, d) = (r', b, d')$, then M' writes the symbol (b, r') on the tape at the current position, moves the head according to d', and enters state r'. For any state q and symbol (a, accept) or (a, reject),

M' halts and enters the corresponding final state. Here, T is the set of transitions in the original Turing machine M, and (q, a) and (r, a, d) represent the symbols (a, q) and (a, r, d) in the modified format for configurations.

Halting Condition

Now, we claim that M halts on input w if and only if the resulting tiling has a finite size. Suppose M halts on input w after t steps. Then the last row of the tiling will consist of tiles representing configurations that are legal and have no outgoing transitions. Therefore, the set of tiles that can appear in the last row is finite, and so is the size of the tiling. Conversely,

suppose the tiling has a finite size t. Then the last row of the tiling can only consist of tiles representing legal configurations that can be reached from the initial configuration in at most t steps. If M does not halt on input w, then there exist legal configurations that can be reached from the initial configuration in arbitrarily many steps. Therefore, the set of tiles that can appear in the last row of the tiling is infinite, which contradicts the fact that the size of the tiling is finite.

Thus, since we have constructed a reduction from $\overline{\text{HALT}}$ to TILE, it follows that TILE is undecidable.

Problem 2 (grade completely)
To prove that for every H, CONTAINS_H is in P, we must show that the total operations reduces to $O(n^k)$ for some constant k. Given our definition of CONTAINS_H, let m be the number of vertices in $H$and n be the number of vertices in G. We want to generate all isomorphic subgraphs of G. First, we must generate a subset of vertices of G of size m. The number of permutations of size m from a set of size n is $\frac{n!}{(n-m)!} = \binom{n}{m} \cdot m!$. It is worth noting that if this value is 0, then $m > n$ and we would reject. We now have our set of m vertices. To check if the induced subgraph by this set of vertices is isomorphic to G, we first let s be the subset of nodes. Going through each pair of nodes $(u, v) \in s$, we check if there exists an edge $(u, v) \in E_G$. We then add this edge to H. In the worst case scenario, going through every edge in G would take $\binom{n}{2} = \frac{n \cdot (n-1)}{2}$ operations and iterating over every pair of vertices would take $\binom{m}{2} = \frac{m \cdot (m-1)}{2}$ operations. Let this induced subgraph be G'. To check if $G'$is equivalent to H, we must iterate over all possible edges which, again in the worst case scenario, is $\binom{m}{2} = \frac{m \cdot (m-1)}{2}$ operations. Thus, the total operations from each step is $O(\binom{n}{m} \cdot m!) \cdot O(n^2) \cdot O(m^2) \cdot O(m^2)$. However, since m is constant, the last two operations can be disregarded. Additionally, we know that $\binom{n}{m} \leq n^m$and$m! \leq m^m$, so the runtime can be reduced from $O(\binom{n}{m} \cdot m!) \cdot O(n^2) \cdot O(m^2) \cdot O(m^2)$ to $O(n^m) \cdot O(n^2) \cdot O(1) \cdot O(1) = O(n^{m+2})$. Since m is a constant, we have shown that CONTAINS_H is in P.

Problem 3

To prove that UNARY SUBSET SUM is in P, we must demonstrate that there exists a deterministic polynomial time algorithm to solve it. We do this by dynamic programming.

Algorithm:
We can create a boolean array, X of size $B + 1$ where each index represents whether a sum j can be reached with a subset of the numbers given. We can first initialize $X[0] =$ true because we can always get a sum of 0 by selecting no elements in the set. Then, we can iterate over each x_i in the array. For each number, we can iterate through the array backwards from B to x_i and set $X[B'] =$ true if $X[B' - x_i]$ is also true. Also, if there is some $x_i = B$, then $X[B] =$ true. After filling the table, we check if$X[B]$is true, and if it is, there exists a subset of numbers that adds to B.

Proof of Correctness:
We prove this by induction.
Base Case: $B' = 0$. The empty set adds up to 0.
Induction Hypothesis: The array X is filled correctly from 0 to $B' - 1$.
Inductive Step: We know there exists a multiset that sums to B' if there also exists a multiset sum equal to $B' - x_i$ by the rule of how the array is set up. Since we know that $X[B' - x_i]$ is correct by the induction hypothesis, it follows that $X[B']$ is also correct. Thus, we have shown that this algorithm is correct.

Time Complexity:
This algorithm relies on two nested loops. The outer loop runs n times (the number of elements) and the inner loop runs B times (the sum to be found). Since B is represented in unary, the input size itself is proportional to B. Therefore, the time complexity of this algorithm is O(nB). Therefore, since it is polynomial with respect to the input size, we have shown that UNARY SUBSET SUM is in P.