CS5110: Complexity Theory

Shreyas Havaldar
CS18BTECH11042

September 27, 2020

1 Exercise Set 8

1.1 To show that 3-SAT $\in$ DSPACE(n)

Our input tape consists a 3-CNF formula, where if m is the number of clauses, we would have $3m$ variable literal symbols and other positions on the input tape would be occupied by either conjunction or disjunction symbols.(Excluding brackets) Let our input size be n. Let the number of variables in our input CNF be k. Thus k is $O(n)$ as the number of variables in our CNF formula cannot exceed the length of the CNF formula itself.

Let there be a Turing Machine TM with a work tape and a counter tape, latter of which has size k.

Thus our counter tape is initialized with value 0 with each of its k bits 0. Thus the size of the counter tape is $O(n)$ as k is $O(n)$. The assignment to each variable is made by using the value on this counter tape, such that the variable x_i is assigned bit on i^{th} position of the counter tape.

The work tape just performs the evaluation of the input CNF according to the current assignment, its size is $O(n)$ as it just requires constant space to save the evaluation values of the current clause and the overall CNF repeatedly.
On input CNF:

1. If the assignment given by the k bits of the counter tape satisfies the formula, return true.
2. If the value on the counter tape is 1 at every position, return false. (We have thus tried all 2^k possible assignments.)
3. Else we still have assignments to explore and thus we increment the value on counter tape by 1 and repeat this process from step 1.

Thus TM evaluates all possible 2^k assignments to the input CNF and accepts if at any point the formula is satisfied, else the input is rejected. Thus we have shown that TM is a deterministic turing machine deciding 3-SAT.

TM is deterministic as we are proceeding in a sequential pre-determined fashion, with no non-determinism at any point.

Total space used by TM is via its two tapes, the work tape and the counter tape, both of which we've shown occupy $O(n)$ space. Thus, 3-SAT $\in$ DPACE(n).

Hence Proved!

1.2 To show that PALINDROME $\in$ DSPACE($\log n$)

If an input x $\in$ PALINDROME, reading the input from both ends until all characters have been read, should return the same string. Let the length of the input be n. If the input x $\in$ PALINDROME, x = $x_{reverse}$, which means $\forall i \in \{0, 1 \dots n-1\}, c_i = c_{n-1-i}$ where c_i represents the character at i^{th} position in the input. We would need atmost $\log n$ bits to store a value from 0 to n using binary notation. Let this number of digits in binary representation of n be k.

Let there be a Turing Machine TM with a counter tape of size k. Our counter tape is initialized with value 0 with each of its k bits 0. Thus the size of the counter tape is $O(\log n)$ as k is $O(\log n)$

On reading the input string we would use L and R tapes to store the read characters from the input string for comparison but that would occupy only constant space so our L and R tapes are of size $O(\log n)$
On input x:

1. Read the value from the counter tape and assign it to i. (0 indexed position)
2. Read c_i, c_{n-1-i} and if $c_i = c_{n-1-i}$ (here c_i represents the character at i^{th} position in the input tape x) increment the value on the counter tape.
3. Else return false.
4. If value on counter tape is equal to $\lfloor(\frac{n-1}{2})\rfloor$, return true.

Thus TM reads the input from both ends, if at any point the characters aren't the same the input is rejected and thus it decides if an input string x $\in$ PALINDROME. Thus we have shown that TM is a deterministic turing machine deciding PALINDROME.

TM is deterministic as we are proceeding in a sequential pre-determined fashion, with no non-determinism at any point.

We need $O(\log n)$ bits to store the index values in our counter tape as shown above. The counter tape, L and R is the only extra space we use which all occupy $O(\log n)$ space. Thus, PALINDROME $\in$ DSPACE($\log n$).

Hence Proved!

1.3 To show that PATH $\in$ NL

PATH = $\{\langle G, s, t\rangle |$ G is a directed graph and has an s-t directed path$\}$.

The input size say m, is $\Omega(n)$ where n is the number of vertices in the graph as irrespective of the method of representation the graph will have to list out its vertices even if there are 0 edges in the graph. So let each vertex be identified by a number from 1 to n. We would need atmost $\log n$ bits to store a value from 1 to n using binary notation.

Let there be a Turing Machine TM with tapes: S,T,V and D. Our S tape is initialized with the source vertex value in $O(\log n)$ space. Our T tape is initialized with the target vertex value in $O(\log n)$ space. Our V tape is initialized with the source vertex value in $O(\log n)$ space and this tape will be the one storing the vertex values during our check to compare them with target vertex value. D is a counter tape that stores the distance from the source vertex we're currently at, it is also the distance of the vertex of value from tape V from the source vertex. Note the shortest path distance from any vertex cannot exceed n-1, as the total number of vertices in the graph itself are n, and any path between two vertices of size more than n-1 must have a cycle in it and thus it would not be a shortest path. Therefore the tape D also takes $O(\log n)$ space using binary notation for the distance. D is initialized with value 0.

Now for an input Graph G,

1. If value read from tape V = value read from tape T, we have found a s-t path thus return true.
2. Else increment the value on tape D, now we are looking at vertices at one distance greater than v from s. If value read from tape D = n, return false as all n-1 length paths have been explored. If value from D is still less than n, non-deterministically guess a vertex from the set of adjacent vertices of v, and then update the value on the tape V to this vertex and repeat from step 1. (v is value read from tape V)

TM thus simulates all possible paths from s to all other vertices without any cycles and if there is an s-t path in the graph it must have been encountered in one of the guesses. If it has not been encountered at any point until the distance from s is n-1, we know that such a path never exists. Thus when the algorithm reaches a point where value on D becomes equal to n-1, we reject the input as no shortest path from s-t can exist. Thus if at any point the guessed vertex value from V is equal to value from T, we accept the input as D is still less than n-1. (Note: If this guessed path contains a cycle, its fine as there will be a shorter path without cycles whose length will be shorter than this and thus shorter than n-1)

Thus TM reads the input graph and non-deterministically determines if a path from s-t exists. Thus TM is a non-deterministic turing machine deciding PATH.

All four tapes that we use, use only $O(\log n)$ space. And as we know input size is $\Omega(n)$, we only use $O(\log m)$ space. Thus, PATH $\in$ NL.

Hence Proved!