The Second Assignment

3190102721 Xu Shengze

Problem 1. Prove that a 4-tape Turing machine working in time $T(n)$ for inputs of length n can be simulated by an ordinary Turing machine working in time $O(T^4(n))$. Bonus points if you design more efficient simulation that requires less than $O(T^4(n))$ time.

Answer 1. Given a 4-tape Turing machine, we can find an ordinary Turing machine that mimics a 4-tape Turing machine to accomplish the same computational task. In the case where the languages accepted by the two Turing machines are the same, we use the ordinary Turing machine to imitate the 4-tape Turing machine, and design the instruction set of the ordinary Turing machine to imitate the calculation process of the 4-tape Turing machine.

We need to simulate the arrangement of the tape. In the one-step calculation of the 4-tape Turing machine, the state is modified once, and the read-write head modifies the characters 4 times at the same time. We use the ordinary Turing machine to imitate the above calculation process of the 4-tape Turing machine, and introduce a special character * in the ordinary Turing machine. The content between the first * and the second * is the content of the first tape, and so on, the content between the fourth * and the fifth * is the content of the fourth tape, so such a tape can simulate the contents of 4 tapes.

Next, we need to simulate the operation of the read-write head. We use one read-write head to simulate the operation of four read-write heads. Introducing special characters into the character set, we use red 1 and black 1 to represent different characters, these two colors have common parts and play the same role in the instruction set. The red 1 marks the position of the read-write head, and the red one is used to indicate the position information of the read-write head.

We use the red 1 to represent the character pointed to by the first read-write head. After reading and writing, go right to the red character position pointed to by the second read-write head, and so on. In this way, we use different font colors to distinguish the characters pointed to by the read and write heads corresponding to different tapes, so as to realize a Turing machine that simulates multiple tapes with one tape.

The price we need to pay for using a normal Turing machine to simulate a 4-tape Turing machine is that the read-write head needs to traverse the tape from left to right to simulate one step of the 4-tape Turing machine. So we need to use an ordinary Turing machine working at time $O(T^4(n))$ to simulate a 4-tape Turing machine working at time $T(n)$.

Bonus In fact, we have the following theorem: for every multi-tape Turing machine working in time $T(n)$ for inputs of length n, there exists an ordinary Turing machine working in time $O(T^2(n))$ equivalent to it. We prove this conclusion as follows.

S uses one of its tapes to represent the contents of all k tapes of M. The tapes are stored consecutively, and the positions of M's read-write heads are marked on the appropriate squares.

Initially, S has its tapes formatted to represent all of M's tapes, and then simulates M's steps. To simulate one step of M, S scans all the information on the tape to determine the symbols under M's read-write head. S then scans the tape again, updating the tape content and read-write head position. If M's read-write head moves to the right to a position on the tape that it has not read before, then S must increase the storage space allocated to this tape. To do this, it shifts part of its tape one space to the right.

Simulating each step of M, S performs two sweeps and may also move to the right up to k times. Each time it takes $O(T(n))$, so, to simulate one-step operation of M, S takes $O(T(n))$ in total.

Now, let's define the total time required for the simulation. At the beginning, S has its bands properly formatted, which takes O(T(n)) steps. Subsequently, S simulates the $T(n)$step operation of M, and each simulation step requires $O(T(n))$steps, so the simulation part requires $T(n) \times O(T(n)) = O(T^2(n))$ steps. Therefore, the entire simulation process requires $O(n) + O(T^2(n))$ steps. Assuming $T(n) > n$ (which is a reasonable assumption, since if the time is less, M cannot even read the input), then the running time of S is $O(T^2(n))$, which proves that complete.

Problem 2. Recall Task 2.7. from the Book that stated the following: Show that any function can be computed by a circuit of depth ≤ 3 with gates of type NOT, AND, and OR, if we allow AND- and OR-gates with arbitrary fan-in and fan-out.

Suppose now that we are limited to bases consisting of 2 logic gates. Show that that any function can be computed by a circuit:

a) of depth ≤ 5 with gates of type NOT and AND if we allow AND-gates with arbitrary fan-in and fan-out.

b) of depth ≤ 5 with gates of type NOT and OR if we allow OR-gates with arbitrary fan-in and fan-out.

Answer 2. We solve this problem using De Morgan's laws.

a) We consider the conjunctive norm form and construct a circuit by using AND-gates with arbitrate fan-in and fan-out. In this case we only need four layers: first layer for negations, second layer for constructing clauses by conjunctions, third layer for negate some clauses, so some clause can equal to clauses in conjunctive normal form, forth layer for conjunctions. If we consider the disjunctive norm form, we'll need need five layers.

b) We consider the disjunctive norm form and construct a circuit by using OR-gates with arbitrate fan-in and fan-out. In this case we only need four layers: first layer for negations, second layer for constructing clauses by disjunctions, third layer for negate some clauses, so some clause can equal to clauses in disjunctive normal form, forth layer for disjunctions. If we consider the conjunctive norm form, we'll need need five layers.

Problem 3. Consider the Subset Sum Problem. Given a set of non-negative integers, and a value sum, determine if there is a subset of the given set with sum equal to given sum.

Example: $\{3, 4, 5, 6, 8\}$ sum=20. Here answer should be positive, since $3+4+5+8 = 20$

Example: $\{2, 3, 7, 8\}$ sum=14. Here answer is negative, we cannot make 14 from numbers that we have.

Show that Subset Sum Problem is in NP.

Answer 3. Here I give two solutions to this question.

Solution 1 Subset Sum problem: given n non-negative integers $w_1, \cdots, w_n$ and a target sum W, the question is to decide if there is a subset $I \subset \{1, \cdots, n\}$ such that $\sum_{i \in I} w_i = W$. Suppose now that a proposed set I is given, the only thing we need to be sure of is whether $\sum_{i \in I} w_i = W$ holds. Adding up at most n numbers, each of size W takes $O(n \log W)$ time, linear in the input size. Hence, Subset Sum Problem is in NP.

Solution 2 The proof process is carried out in two steps. (1) Solution polynomial sized. One of the solutions to the subset sum problem can be a set S at index i from set $[n]$ that corresponds to $A_i s$ whose sum is number T. So the size of the solution could be at most n. So it is polynomial in the input size. (2) Polynomial time verification. If we have the set S, we can verify the solution by adding the corresponding $A_i s$ and comparing this sum with T. In this process, the number of additions is at most $n - 1$. So addition and comparison can be done in polynomial time. Therefore, Subset Sum Problem is in NP.

Problem 4. Consider TQBF (True quantified Boolean formula) problem that we discussed in lecture 7. For each formula, verify whether it is true or false:

a) $\forall x \exists y \exists z((x \vee y) \wedge z)$

b) $\forall x \exists y((x \vee y) \wedge (\neg x \vee \neg y))$

c) $\forall x(x)$

d) $\forall x \forall y \exists z((x \wedge z) \vee y)$

Answer 4.

a) This formula is a true QBF. $\forall x \exists y \exists z((x \vee y) \wedge z) = \forall x \exists y \exists z((x \wedge z) \vee (y \wedge z))$. For whether x is true or false, we just need to take y and z to be true, then the formula is always true.

b) This formula is a true QBF. For y is an existential variable, when x is true, we take y to be false, when x is false, we take y to be true, then the formula is always true.

c) This formula is a false QBF. The formula is false when we take x to be false. We cannot guarantee that the formula is always true.

d) This formula is a false QBF. $\forall x \forall y \exists z((x \wedge z) \vee y) = \forall x \forall y \exists z((x \vee y) \wedge (z \vee y))$. When we take x to be false and y to be false, whether the value of z is true or false, the formula is false.