Notes on NP-Completeness

(Revised 4/8/04)

This is a summary of what we did in class on the topic of NP-completeness.

1 Circuits

First we introduced the definition of Boolean circuit, as in Sipser Section 9.3. We observed the following result

Lemma 1 *Let $f : \{0,1\}^n \to \{0,1\}$ be an arbitrary Boolean function. Then there is a circuit of size $O(2^n)$ that computes f.*

Proof: We give a recursive construction of the circuit. If $n = 1$, then either $f(x) = x$, in which case it is computed by a circuit of zero gates, or $f(x) = \neg x$, which can be computed by a circuit of size one, or $f(x) = 0 = (x \wedge \neg x)$, which is computed by a circuit of size two, or $f(x) = 1 = (x \vee \neg x)$, which is also computed by a circuit of size 2.

Let now f be an arbitrary function of n variables. We can write it as

$$f(x_1, \ldots, x_n) = (x_n \wedge f(x_1, \ldots, x_{n-1}, 1)) \vee (\neg x_n \wedge f(x_1, \ldots, x_{n-1}, 0))$$

Where both $f(x_1, \ldots, x_{n-1}, 1)$ and $f(x_1, \ldots, x_{n-1}, 0)$ are functions of $n-1$ variables, that can be recursively realized by a circuit.

The size $S(n)$ of the circuit constructed this way satisfies the recursion $S(1) \leq 2$, $S(n) \leq 4 + 2S(n-1)$, which solves to $S(n) \leq 3 \cdot 2^n - 4$. $\Box$

Then we proved the following result, which is Sipser Theorem 9.25.

Theorem 2 *Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$ be a Turing machine that, on inputs of length n, runs in time at most t. Then there is a circuit of size $O((|\Gamma| \cdot |Q|)^3 \cdot t^2)$ that, given an input x of length n, outputs 1 if and only if M accepts x.*

Proof: We first construct a circuit C_{next} that, given a configuration of M that uses $\leq t$ cells of tape, computes the configuration at the following step. A final configuration is left unchanged.

Let M have tape alphabet Γ and set of states Q. We represent a configuration by using t blocks of bits. The i-th block of bits contains the alphabet element of the i-th cell of the tape (represented as a sequence of $\lceil \log |\Gamma| \rceil$ bits), a bit that says whether the head of the machine is over the i-th cell of the tape, and, if so, the current state of the machine, represented as a sequence of $\lceil \log |Q| \rceil$ bits. Each block, therefore, is $1 + \lceil \log |\Gamma| \rceil + \lceil \log |Q| \rceil$ bits long. Let us call this number B. (Note that, for a fixed machine M, B is a constant.)

We want to build a circuit C_{next} that, given ct bits in input representing a configuration, produces ct bits in output representing the next-step configuration.[1] It suffices to observe that

[1] There is one more detail to take care of: what happens if the input is a configuration c that uses t cells of tape and the next-step configuration c' uses $t+1$ cells of tape? In this case, we will let the circuit output only the content of the first t cells of the tape of c'.

every bit of the output depends on only $\leq 3B$ bits of the input, and so each output bit of the circuit can be computed using $O(2^{3B})$ gates, so that the entire circuit has size $O(t \cdot 2^{3B})$. To justify the previous observation, let c be an input configuration for the circuit and c' be the desired output. The portion of c' corresponding to the i-th cell of the tape depends only on the portion of c corresponding to the $(i-1)$-th, i-th, and $(i+1)$-th cells of the tape; in one step, the content of no other cell can have any effect on the i-th cell. In total, these three cells are described by $3B$ bits, including a description of where the head of the machine is and what is the state.

Let us now construct a circuit C_t by layering t copies of circuit C_{next} one on top of the other, that is, with the outputs of the i-th copy fed as inputs of the $(i+1)$-th copy. Clearly, C_t has size $O(t^2 \cdot 2^{3c})$ and, given a configuration c, $C_t(c)$ computes the configuration reached by M starting from c in t steps. We can modify it into a circuit C'_t of size $O(t^2 \cdot 2^{3c})$ that has only one output and such that $C'_t(c) = 1$ if and only if M reaches an accepting configuration starting from c and running for at most t steps.

Finally, let us hard-wire into C'_t that the head is in the first cell, that the state is q_0, and that all the cells except the first n contain a blank symbol, and let us call C the resulting circuit. Now, on input x, $C(x) = 1$ if and only if M accepts x in at most t steps.[2] $\Box$

2 Satisfiability Problems

Definition 1 (Circuit-SAT) *Define the Circuit Satisfiability (Circuit-SAT) problem as follows: given a circuit C the question is whether there is an input x such that $C(x) = 1$.*

Using Theorem 2 it is easy to prove that Circuit Satisfiability is NP-complete.

Theorem 3 *Circuit-SAT is NP-complete.*

PROOF: This is Theorem 9.26 in Sipser's book, see the proof there. $\Box$

Next we define the problem 3SAT. In 3SAT, an input is a Boolean formula in 3-Conjunctive-Normal-Form (3CNF). A 3CNF formula is a AND-of-ORs, with each OR being over precisely three distinct variables. Variables are allowed to be completed.

Definition 2 (3SAT) *The 3SAT problem is: given a 3CNF formula ϕ, is there an assignment of values to the variables that satisfies ϕ?*

It is easy to see that 3SAT is in NP.

Theorem 4 *Circuit-SAT $\leq_m^p$ 3SAT. Therefore 3SAT is NP-hard, and so NP-complete.*

This is Theorem 9.27 in Sipser's book.

[2] There is one final detail: C_t and C'_t expect in input a configuration, which is a sequence of triples (b, q, a) where b is a bit that tells whether the head is on that cell of the tape, q tells, if $b = 1$, what is the state of the machine, and a is the *tape alphabet* element on that cell of the tape. After we hard-wire the values of b and q, we still cannot let a be the input of the circuit, because each a is a sequence of $\lceil \log_2 |\Gamma| \rceil$ bits designed to represent an element of Γ, while we want our final circuit to have only one input bit per cell of the tape. We can solve this problem by assuming that Γ is represented in binary so that 0 is mapped into $0 \cdots 00$ and 1 is mapped into $0 \cdots 01$, then we just have to hardwire zeroes into all the input bits corresponding to an alphabet element except for the last bit in each cell.

3 Graph Problems

Definition 3 *In an undirected graph $G = (V, E)$:*

- *A* clique *is a subset $K \subseteq V$ such that all the pairs $u, v \in K$ are connected by an edge in E.*
- *An* independent set *is a subset $S \subseteq V$ such that no pair $u, v \in S$ is connected by an edge in E.*
- *A* vertex cover *is a subset $C \subseteq V$ such that for every edge $(u, v) \in E$ at least one of u or v is an element of C.*

Definition 4 (Clique) *The Clique problem is: given a graph G and an integer k, is there a clique of size at least k?*

Definition 5 (IS) *The Independent Set (IS) problem is: given a graph G and an integer k, is there an independent set of size at least k?*

Definition 6 (VC) *The Vertex Cover (VC) problem is: given a graph G and an integer k, is there a vertex cover of size at most k?*

It is easy to see that Clique, IS and VC are all NP problems.

Theorem 5 *3SAT $\leq_m^p$ IS. Therefore IS is NP-complete.*

PROOF: Let ϕ be a formula with m clauses and n variables, and with precisely three distinct literals in each clause.

We construct a graph $G = (V, E)$ as follows. The graph has $3m$ vertices, three for each clause. The three vertices corresponding to a clause form a triangle and each of them is labeled with one of the literals in the clause. So, if, for example, the variable x_i occurrs three times positively and twice negated in ϕ, then the graph G will have three vertices labeled x_i and two vertices labeled $\bar{x}_i$. Then we add an edge between any two vertices that are labeled by complementary literals. This completes the construction of the graph G.

Claim 6 *If ϕ is satisfiable, then G has an independent set of size m.*

PROOF: Consider a satisfying assignment for ϕ; then in each clause C of ϕ there is at least a literal that is made true by the assignment. Let v_C be the vertex in G that corresponds to the first literal in C that is satisfied by the assingment. Consider the set of m vertices $S = \{v_C : C \in \phi\}$, and let us see that it is an independent set. There are only two types of edges in the graph: edges between the three vertices corresponding to a clause (let us call them "triangle edges") and edges between complementary literals (let us call them "consistency edges"). The vertices in S belong to distinct triangles, so they do not share any triangle edge; furthermore, their labels are all consistent with a fixed assignment, and so they cannot share any consistency edge. So S is an independent set of size m. □

Claim 7 *If G has an independent set of size m, then ϕ is satisfiable.*

PROOF: Let S be an independent set of size m. For every variable x_i in ϕ, set x_i to TRUE if there is a vertex labeled x_i in S, and set x_i to FALSE if there is a vertex labeled $\bar{x}_i$ in S (note that, since S is an independent set, S cannot contain both types of vertices). If neither type of vertices is in S, set x_i to FALSE. Let us call this assignment a.

Since S is an independent set, it cannot contain more than one vertex from each triangle, and since there are only m triangles in the graph it means that S contains precisely one vertex from each triangle.

Let now C be any clause in ϕ. One of the vertices of the triangle corresponding to C is in S, and so a satisfies one of the literals of C, and so C is satisfied by a. This is true for every clause C, so ϕ is satisfied by a. $\Box$

The reduction from 3SAT to IS is then the mapping of ϕ into the pair (G, m). $\Box$

Theorem 8 *IS $\leq_m^p$ VC. Therefore VC is NP-complete.*

PROOF: In a graph $G = (V, E)$, a set S is an independent set if and only if $V - S$ is a vertex cover. (This statement follows easily from the definitions.) Here is the reduction from IS to VC: on input (G, k), output $(G, |V| - k)$. $\Box$

Theorem 9 *IS $\leq_m^p$ Clique. Therefore Clique is NP-complete.*

PROOF: If $G = (V, E)$ is a graph, define the *complement* of G to be the graph $\bar{G} = (V, \bar{E})$ with the same set of vertices and with precisely those edges that are not in G. In a graph $G = (V, E)$, a set S is an independent set if and only if S is a clique in $\bar{G}$. (This statement follows again easily from the definitions.) Here is the reduction from IS to Clique : on input (G, k), output $(\bar{G}, k)$. $\Box$

4 Problems About Subsets of Integers

Definition 7 (Subset Sum) *In Subset Sum the input is a sequence of non-negative integers $a_1, \ldots, a_n$ and a target value t. The question is whether there is a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = t$.*

You may think of the subset sum problem as the problem of giving exact change using coins of various denominations. There is an algorithm for this problem that uses dynamic programming and that runs in time polynomial in n and t. Since the size of the input is roughly $n \log t$, this running time may not be polynomial in the length of the input. We show that the problem is NP-complete and that a polynomial time algorithm is unlikely to exist. By the way, notice that the problem is easily seen to be in NP.

Theorem 10 *VC $\leq_m^p$ Subset Sum. Therefore Subset Sum is NP-complete.*

PROOF: Sipser's book has a different reduction in Theorem 7.37.

In the reduction from VC we start from a graph $G = (V, E)$ with n vertices and m edges and a parameter k, and we define an instance of Subset Sum witn $n + m$ integers, an integer a_v for every vertex $v \in V$ and an integer b_e for every edge $e \in E$. We write each integer in base 4, with a digit for each edge in E. For $j = 0, \ldots, |E| - 1$, let $e = (u, v)$ be the j-th edge of E, then the integers a_u, a_v and $b_{u,v}$ have a 1 as their j-th digit in base 4, and all other integers have a zero. The integers a_v, in addition, have a 1 as their $|E|$-th digit, while the integers $b_{u,v}$ have a zero.

We set the target to be $t = k \cdot 4^{|E|} + \sum_{j=0}^{|E|-1} 2 \cdot 4^j$.

Suppose that there is a vertex cover in G with at most k vertices. Then, in particular, there is a vertex cover C with precisely k vertices. (Adding vertices to a vertex cover maintains the property of being a vertex cover.) Consider the set of integers that consists of all the a_v such that $v \in C$ and all the $b_{u,v}$ such that precisely one of u or v is in C. If we sum these integers, and we write the sum in base 4, we will see that all the first $|E|$ digits are 2. Each of the first $|E|$ digits corresponds to an edge (u, v) in the graph, and only three integers have a 1 in that digit: a_u, a_v and $b_{u,v}$; by our choice, precisely two of these integers are part of our sum, and so we get a 2, with no carry, in that digit. Finally, we sum k integers of the form a_v, and so the total has a leading term $k \cdot 4^{|E|}$.

For the other direction, suppose that there are subsets $S \subseteq V$ and $T \subseteq V$ such that $\sum_{u \in S} a_u + \sum_{(u,v) \in T} b_{u,v} = t$. We want to claim that S is a vertex cover of size at most k. First of all, it is clear that $|S| \leq k$, because each a_u is bigger than $4^{|E|}$ but $t < (k + 1) \cdot 4^{|E|}$. It remains to prove that it is a vertex cover. When we compute the sum of the integers in base 4, we note that we never have a carry in the first $|E|$ digits, because, in each digit, there are only three integers in our instance that have a 1 in that digit, and all others have a zero. If no carry occurrs, and if we have a 2 in each of the first $|E|$ digits, it means that, for each edge (u, v), at least one of a_u or a_v participates in the summation, and so at least one of u or v is in S, and so S is a vertex cover. $\Box$

We define three more problems: Partition, Knapsack and Bin Packing.

Definition 8 (Partition) *In the Partition problem we are given integers $a_1, \ldots, a_n$ and the question is whether there is a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = \sum_{j \notin S} a_j$.*

Definition 9 (Knapsack) *In the Knapsack problem we are given integer costs $c_1, \ldots, c_n$ and volumes $v_1, \ldots, v_n$, a cost target t and a volume bound B. The question is whether there is a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} c_i \geq t$ and $\sum_i v_i \leq B$.*

The interpretation of Knapsack is that we have n objects, and object i has volume v_i and cost c_i. We have a bag of size B and we want to fit objects of maximum total cost into the bag. In the decision version, we want to know if it is possible to fit objects of total cost at least t into the bag. The problem is solvable in time polynomial in n, t and B using dynamic programming. The size of the problem is about $n(\log t + \log B)$, and so the dynamic programming algorithm may be exponential in the input length if the integers are very large compared to n.

Definition 10 (Bin Packing) *In the Bin Packing problem we are given volumes $v_1, \ldots, v_n$, a volume bound B, and a target k. The question is whether we can partion the integers $v_1, \ldots, v_n$ into k subsets such that the integers in each subset sum to at most B.*

The interpretation of Bin Packing is that we have n objects, with object i having volume v_i. We are given k bins, each of size B, and we want to fit all the objects into the k bins.

It is easy to see that Partition, Knapsack and Bin Packing are in NP.

Theorem 11 *Subset Sum $\leq_m^p$ Knapsack. Therefore Knapsack is NP-complete.*

PROOF: Start from an instance $a_1, \ldots, a_n, t$ of Subset Sum. Define an instance of Knapsack with n items by setting $c_i = v_i = a_i$, that is, the cost and the volume of object i are precisely the integer a_i. Then set $t = B = k$. This is clearly a YES-instance of Knapsack if and only $(a_1, \ldots, a_n, t)$ is a YES-instance of Subset Sum. $\Box$

Theorem 12 *Subset Sum* $\leq_m^p$ *Partition. Therefore Partition is NP-complete.*

PROOF: Given an instance $I = (a_1, \ldots, a_n, t)$ of Subset Sum, define $A = \sum_{i=1}^n a_i$ to be the total sum of the integers. Consider the instance of Partition $I' = (a_1, \ldots, a_n, a_{n+1}, a_{n+2})$ where $a_{n+1} = 2A - t$ and $a_{n+2} = A + t$. The total sum of the integers in I' is $4A$, so I' is a YES-instance of Partition if and only if there is a subset of the integers that sums to $2A$.

If I is a YES-instance of Subset Sum, let $S \subseteq \{1, \ldots, n\}$ be the solution such that $\sum_{i \in S} a_i = t$. Then $S \cup \{a_{n+1}\}$ is a solution that shows that I' is a YES-instance of Partition.

If I' is a YES-instance of Partition, let $S \subseteq \{1, \ldots, n+2\}$ be the solution such that $\sum_{i \in S} a_i = 2A$. Clearly, precisely one of $n+1$ or $n+2$ belongs to S. If $(n+1) \in S$, then $S - \{n+1\}$ is a solution that proves that I is a YES-instance of Subset Sum. Otherwise, if $n+2 \in S$, then $\sum_{i \in S, i \leq n} a_i = 2A - a_{n+2} = A - t$, from which we deduce that $\sum_{i \notin S, i \leq n} a_i = t$ and the set $S' = \{i : i \leq n, i \notin S\}$ proves that I is a YES-instance of Subset Sum. $\Box$

Theorem 13 *Partition* $\leq_m^p$ *Bin Packing. Therefore Bin Packing is NP-complete.*

PROOF: Given an instance $a_1, \ldots, a_n$ of Partition, create an instance of Bin Packing by setting $v_i = a_i$, $B = (\sum_{i=1}^n a_i)/2$ and $k = 2$. $\Box$