Introduction to Algorithms	Problem Set 7
CS 4820, Spring 2014	Due 11:59pm Friday, March 21

Hand in your solutions electronically using CMS. Each solution should be submitted as a separate file. For multi-part problems, all parts of the solution to that problem should be included in a single file.

Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.

The following problems ask you to design "self-contained polynomial-time reductions." You should interpret that term as follows. If* A *and* B *are two decision problems, a self-contained polynomial-time reduction from* A *to* B *is a polynomial-time algorithm that takes an instance of* A *and outputs an instance of* B*. To show that the reduction is correct, you must prove two things. First, that it transforms* YES *instances of* A *to* YES *instances of* B*. Second, that it transforms* NO *instances of* A *to* NO *instances of* B*.

(1) *(10 points)* In class, we stated without proof the important fact that 3SAT is NP-hard, which means that every problem in NP reduces to 3SAT in polynomial time.

To gain some intuition about how these reductions work, this exercise asks you to give a self-contained polynomial-time reduction from INDEPENDENT SET to 3SAT.

Solution

Given an instance of INDEPENDENT SET that asks whether a graph $G = (V, E)$ has an independent set of size k, our reduction creates an instance of 3SAT having $n+(n+1)(k+1)+2$ Boolean variables, where n is the number of vertices of G. Henceforth we will identify the vertex set V with the set $[n] = \{1, \dots, n\}$. We will use two sets of variables $\{x_i \mid i = 1, \dots, n\}$ and $\{y_{ij} \mid i = 0, \dots, n;\ j = 0, \dots, k\}$ plus extra variables w, z. The variables will have the following interpretations; these interpretations are not technically part of the reduction, but they aid in understanding the solution.

1. x_i is true if and only if vertex i belongs to the independent set.
2. y_{ij} is true if and only if at least j of the vertices in the set $\{1, \dots, i\}$ belongs to the independent set.
3. w, z are dummy variables whose purpose is to convert clauses with fewer than three literals into clauses with exactly three literals.

Before presenting the rest of the reduction, let us elaborate on the last point: the use of the variables w, z. Any disjunction of two Boolean variables, $a \vee b$, can be rewritten in the equivalent form $(a \vee b \vee z) \wedge (a \vee b \vee \bar{z})$. This is equivalent in the sense that the following two facts hold:

- any truth assignment to a, b that satisfies $a \vee b$ also satisfies $(a \vee b \vee z) \wedge (a \vee b \vee \bar{z})$, regardless of the truth value assigned to z;
- any truth assignment to a, b, z that satisfies $(a \vee b \vee z) \wedge (a \vee b \vee \bar{z})$ also satisfies $a \vee b$.

Similarly, a disjunction with just one literal (i.e., an assertion that a Boolean literal, a, is true) is equivalent to the following conjunction of four clauses.

$$(a \vee w \vee z) \wedge (a \vee \bar{w} \vee z) \wedge (a \vee w \vee \bar{z}) \wedge (a \vee \bar{w} \vee \bar{z}).$$

Now we can continue with the description of our reduction, presenting the constraints that define the independent set problem and their translations into Boolean disjunctions with three variables each.

1. If $\{i, i'\}$ is an edge of G then the independent set cannot contain both i and i'. In other words, at least one of $x_i, x_{i'}$ must be false. Equivalently:

$$\forall\{i, i'\} \in E \; (\bar{x}_i \vee \bar{x}_{i'} \vee z) \wedge (\bar{x}_i \vee \bar{x}_{i'} \vee \bar{z}).$$

2. The independent set contains at least k vertices. In other words, y_{nk} must be true. Equivalently:

$$(y_{nk} \vee w \vee z) \wedge (y_{nk} \vee \bar{w} \vee z) \wedge (y_{nk} \vee w \vee \bar{z}) \wedge (y_{nk} \vee \bar{w} \vee \bar{z}).$$

3. Now we need to add some constraints to ensure that y_{ij} is not set to true unless at least j of the variables $x_1, \ldots, x_i$ are true. The case $i = 0$ is handled differently from $i > 0$.
   (a) The variable y_{0j} is false for $j > 0$. Equivalently:

$$\forall j > 0 \; (\bar{y}_{0j} \vee w \vee z) \wedge (\bar{y}_{0j} \vee \bar{w} \vee z) \wedge (\bar{y}_{0j} \vee w \vee \bar{z}) \wedge (\bar{y}_{0j} \vee \bar{w} \vee \bar{z}).$$

   (b) For $i > 0, j > 0$, the variable y_{ij} is not true unless $y_{i-1,j-1}$ is also true, and in addition at least one of $y_{i-1,j}$ or x_i is true. Equivalently:

$$\forall i > 0 \; \forall j > 0 \; (\bar{y}_{ij} \vee y_{i-1,j-1} \vee z) \wedge (\bar{y}_{ij} \vee y_{i-1,j-1} \vee \bar{z}) \wedge (\bar{y}_{ij} \vee y_{i-1,j} \vee x_i).$$

Our reduction creates the conjunction of all of the clauses listed above. Letting m denote the number of edges of G, there are $2m + 4 + 4k + 3nk$ clauses, so the running time of the reduction is $O(m + nk)$, which is polynomial in the size of the INDEPENDENT SET instance. (Note that $k \leq n$ in an instance of INDEPENDENT SET, so that although the input to the INDEPENDENT SET problem expresses the value of k in binary, a running time that depends linearly on k is still considered polynomial.)

To prove the correctness of the reduction, we must show two things.

1. If the graph G contains an independent set of size k, then there is a truth assignment of the Boolean variables that satisfies all of the clauses. The discussion that accompanied our description of the reduction already reveals what the truth assignment should be. Namely, let S be an independent set of size k and set x_i to true if and only if i belongs to S. Then set y_{ij} to true if and only if S contains at least j elements of $\{1, \ldots, i\}$, interpreting the latter set as the empty set in the case $i = 0$. Finally, we set arbitrary truth values for w, z; for concreteness we set both of them to true. Now we must check that this assignment satisfies all of the clauses listed above. The first type of clause is satisfied because S never contains both endpoints of an edge. The second type of clause is satisfied because y_{nk} is true, by our assumption that S has at least k elements. The third and final type of clause is satisfied because y_{0j} is false for every $j > 0$ (i.e., the intersection of S with the empty set has zero elements) and y_{ij} is false unless both $y_{i-1,j-1}$ and $y_{i-1,j} \vee x_i$ are true (i.e., the only ways for S to contain j elements of $\{1, \ldots, i\}$ are if it already contains j elements of $\{1, \ldots, i-1\}$, or else if it contains $j - 1$ elements of that set and also contains i). Thus, we have verified that there is a truth assignment that satisfies all of the clauses in the event that G has an independent set of size k.

2. If there is a truth assignment of the Boolean variables that satisfies all of the clauses, then G contains an independent set of size k. Given a truth assignment that satisfies all of the clauses, let S denote the set of all i such that x_i is true. The first set of clauses ensures that S is an independent set: if $\{i, i'\}$ were an edge such that x_i and $x_{i'}$ were both true, then no truth assignment for z could succeed in satisfying both of the clauses $(\bar{x}_i \vee \bar{x}_{i'} \vee z) \wedge (\bar{x}_i \vee \bar{x}_{i'} \vee \bar{z})$. It remains to show that $|S| \geq k$. To do so, we will prove that for all $i \geq 0$, $j > 0$, if y_{ij} is true then $|S \cap \{1, \ldots, i\}| \geq j$. In the base case $i = 0$, this induction hypothesis holds because y_{0j} is false for all $j > 0$. Indeed, if y_{0j} were true for some j, there would be no truth assignment of w, z that simultaneously satisfies all four of the clauses in $(\bar{y}_{0j} \vee w \vee z) \wedge (\bar{y}_{0j} \vee \bar{w} \vee z) \wedge (\bar{y}_{0j} \vee w \vee \bar{z}) \wedge (\bar{y}_{0j} \vee \bar{w} \vee \bar{z})$. For the induction step, first note that the induction hypothesis is trivially satisfied if y_{ij} is false. On the other hand, if y_{ij} is true, then the clause $\bar{y}_{ij} \vee y_{i-1,j} \vee x_i$ can only be satisfied if either $y_{i-1,j}$ or x_i is true. In the former case, our induction hypothesis implies that $|S \cap \{1, \dots, i-1\}| \geq j$, from which we can conclude that $|S \cap \{1, \dots, i\}| \geq j$ as well. In the latter case, x_i is true so $i \in S$. Furthermore, the pair of clauses $(\bar{y}_{ij} \vee y_{i-1,j-1} \vee z) \wedge (\bar{y}_{ij} \vee y_{i-1,j-1} \vee \bar{z})$, combined with our assumption that y_{ij} is true, implies that $y_{i-1,j-1}$ must be true, as otherwise no truth assignment for z could lead to both clauses being satisfied. By the induction hypothesis, this means that $|S \cap \{1, \dots, i-1\}| \geq j-1$. Combined with the fact that $i \in S$, we again deduce that $|S \cap \{1, \dots, i\}| \geq j$ which completes the justification of this induction step.

Finally, the four clauses $(y_{nk} \vee w \vee z) \wedge (y_{nk} \vee \bar{w} \vee z) \wedge (y_{nk} \vee w \vee \bar{z}) \wedge (y_{nk} \vee \bar{w} \vee \bar{z})$ imply that y_{nk} is true, as otherwise no truth assignment of w, z could simultaneously satisfy all four of the clauses. From the inductive argument in the preceding paragraph, we know that the fact that y_{nk} is true implies that $|S \cap \{1, \dots, n\}| \geq k$, i.e. the independent set S has at least k elements, as desired.

(2) *(20 points)* If G is a flow network and f is a flow in G, we say that f *saturates* an edge e if the flow value on that edge is equal to its capacity, i.e. $f(e) = c_e$. The FLOW SATURATION problem is the following decision problem: given a flow network G and a positive integer k, determine if there exists a flow f in G such that f saturates at least k edges of G. This exercise asks you to prove that FLOW SATURATION is NP-complete.

(a) *(2 points)* Show that FLOW SATURATION is in NP.

Solution

A polynomial-time verifier for FLOW SATURATION works as follows. Given an instance of FLOW SATURATION specified by a flow network G and a parameter k, and given a flow f, the verifier first checks that f satisfies the flow capacity and conservation constraints ($O(m)$ time) and then it tests every edge to see whether f saturates the edge ($O(m)$ time). The verifier outputs "yes" if and only if f satisfies the flow capacity and conservation constraints and saturates k or more edges.

(b) *(8 points)* Give a self-contained polynomial-time reduction from SUBSET SUM to FLOW SATURATION.

Solution

Given an instance of SUBSET SUM specified by positive integers $n_1, \dots, n_m$ and a target sum W, our reduction calculates the number $S = n_1 + \cdots + n_m$. If $S < W$ then the input is a "no" instance of SUBSET SUM and the reduction creates an arbitrary "no" instance of FLOW SATURATION, for example a flow network with two vertices and no edges, accompanied by the parameter $k = 1$. If $S \geq W$ then our reduction constructs a flow network G with the following vertices and edges.

- Source s, sink t.
- Vertices $v_1, \dots, v_m$, collectively called "Level 1."
- Vertices a, b, collectively called "Level 2."
- Edges (s, v_i) of capacity n_i for each vertex v_i in Level 1.
- Edges (v_i, a) and (v_i, b) of capacity n_i for each $i = 1, \dots, m$.
- Edges (a, t) and (b, t) of capacities W and $S - W$, respectively.

The instance of FLOW SATURATION constructed by our reduction consists of the network G and the parameter $k = 2m + 2$. Our network has $m + 4$ vertices and $3m + 2$ edges. Constructing an edge with capacity c requires writing $O(\log c)$ bits, so the running time of our reduction is $O(\sum_{i=1}^{m}(1 + \log n_i))$, which is linear in the input size, implying that the reduction runs in polynomial time.

Let G denote the flow network produced by our reduction. To prove correctness of the reduction, we need to show two things.

1. If there is a subset of the numbers that sums up to W, then there is a flow in G that saturates at least $2m + 2$ edges. First, note that if there is a subset of the numbers that sums up to W then $W \leq S$, so we are in the non-trivial case of the reduction, i.e. the case that produces a flow network with non-empty edge set. Letting J denote a subset of $[m]$ such that $\sum_{i \in J} n_i = W$, we design a flow in G as follows: the flow sends n_i units from s to v_i for each i, then v_i sends n_i units on edge (v_i, a) if $i \in J$ and on edge (v_i, b) otherwise. Finally, edges (a, t) and (b, t) sends W and $S - W$ units of flow, respectively. The constructed flow clearly satisfies the capacity constraints and it clearly satisfies flow conservation at v_i for all i. Furthermore, the incoming flow at a is equal to $\sum_{i \in J} n_i$, which equals W by our assumption about the set J. Since the outgoing flow at a is also equal to W, this verifies flow conservation at a. Similarly, $\sum_{i \notin J} n_i = S - W$ and this verifies flow conservation at b. Finally, the number of saturated edges is $2m + 2$ because each v_i is incident to two saturated edge (and no edge is counted twice among these $2m$) and additionally the edges (a, t) and (b, t) are saturated.

2. If there is a flow in G that saturates at least $k = 2m + 2$ edges, then there is a subset of the numbers that sums up to W. First, note that if there is a f flow in G that saturates at least k edges, then G has a non-zero number of edges so we are in the case $W \leq S$. Now consider any vertex v_i in Level 1. The only incoming edge to v_i has capacity n_i, and both outgoing edges of v_i have capacity n_i as well. By the flow capacity and conservation constraints at vertex v_i, it follows that at most one of the outgoing edges of v_i can be saturated, and that the only way to saturate one of those outgoing edges is to send n_i units of flow on that edge and 0 on the other outgoing edge. Thus, it is impossible to saturate more than m of the edges leaving Level 1. Since the entire network G contains only $m + 2$ edges other than those leaving Level 1 (m edges leaving s and 2 edges entering t), we may conclude that the only way f can saturate at least $2m + 2$ edges of G is by saturating all of the edges leaving s and entering t ($m + 2$ of them) and additionally saturating m of the edges leaving Level 1. Furthermore, as observed earlier, to saturate m of the edges leaving Level 1, it must be the case that each vertex v_i sends n_i units of flow on one of its outgoing edges and 0 units on the other one. Now, let J denote the set of all i such that v_i sends n_i units of flow on edge (v_i, a). The amount of incoming flow at vertex a is equal to $\sum_{i \in J} n_i$ and the amount of outgoing flow is equal to W, so by flow conservation it must be the case that $\sum_{i \in J} n_i = W$. Thus, there is a subset of the numbers that sums up to W, as claimed.

(c) *(10 points)* Give a self-contained polynomial-time reduction from INDEPENDENT SET to FLOW SATURATION.

Solution

Given an instance of INDEPENDENT SET specified by a graph G and a parameter k, our reduction works as follows. Let n, m denote the number of vertices and edges in G. Equate the vertex set of G with the set $[n]$, and for $i = 1, \ldots, n$ let d_i denote the degree of vertex i. Number the edges of G as $\{e_1, \ldots, e_m\}$. Our reduction constructs a flow network with the following vertices and edges.

- Source s, sink t.
- Vertices $v_1, \ldots, v_n$, collectively called "Level 1."
- Vertices $w_1, \ldots, w_m$, collectively called "Level 2."
- Edges (s, v_i) of capacity d_i for $i = 1, \ldots, n$.
- Edges (v_i, w_j) of capacity 1 for each pair i, j such that i is an endpoint of e_j.
- Edges (w_j, t) of capacity 1 for $j = 1, \ldots, m$.

The instance of FLOW SATURATION constructed by our reduction consists of the network G and the parameter $k' = k + 2m$. Our network has $n + m + 2$ vertices and $n + 3m$ edges. Constructing an edge with capacity c requires writing $O(\log c)$ bits; our graph has n edges of capacity at most n and $3m$ edges of capacity 1, so the running time of our reduction is $O(n \log n + m)$. In particular, the reduction runs in polynomial time.

Let G' denote the flow network produced by our reduction. To prove correctness of the reduction, we need to show two things.

1. If G has an independent set of size k, then G' has a flow that saturates at least k' edges. Let S be an independent set of size k in G. For every edge e_j, choose one endpoint $i(j)$ as follows: if e_j has an endpoint that belongs to S, then there must be only one such endpoint and we let $i(j)$ be that endpoint; otherwise select $i(j)$ to be an arbitrary endpoint of e_j. Now construct a flow f in G' as follows. For every $j = 1, \ldots, m$, send one unit of flow from $v_{i(j)}$ to w_j. Send zero flow on all other edges between Level 1 and Level 2. Send one unit of flow on every edge into t. Finally, for every $i = 1, \ldots, n$, the flow on edge (s, v_i) is set equal to the number of edges e_j such that $i(j) = i$. Note that this is also equal to the number of units of flow leaving v_i, and that it is bounded above by d_i, hence flow conservation is satisfied at v_i and the capacity constraint of edge (s, v_i) is satisfied. The remaining flow conservation and capacity constraints are trivial to check, given the way we have defined the flow f. Finally, by construction f saturates the following $k' = k + 2m$ edges: the edges (s, v_i) for all $i \in S$, the edges $(v_{i(j)}, w_j)$ and (w_j, t) for $j = 1, \ldots, m$. It is trivial to see that $(v_{i(j)}, w_j)$ and (w_j, t) are saturated, and to verify that (s, v_i) is saturated for $i \in S$ we observe that when $i \in S$, every edge e_j incident to i chooses $i(j) = i$.

2. If G' has a flow that saturates at least k' edges then G has an independent set of size k. For $j = 1, \ldots, m$ the only outgoing edge of w_j has capacity 1, and both of the incoming edges of w_j also have capacity 1, so at most one of the incoming edges of w_j can be saturated by a flow. This means that every flow saturates at most $2m$ of the edges incident to Level 2 (all m of the edges leaving the level, and m of the edges entering the level). So if a flow succeeds in saturating $k' = k + 2m$ edges, it must saturate at least k of the edges from s to Level 1. Let S denote the set of all i such that edge (s, v_i) is saturated. We have argued that $|S| \geq k$, now let us show that S is an independent set in G. If e_j is any edge of G, let i, i' denote the endpoints of e_j. We must prove at least one of i, i' does not belong to S. As noted earlier, at most one of the incoming edges w_j can be saturated. Assume without loss of generality that edge (v_i, w_j) is not saturated. Then not all of the outgoing edges of v_i are saturated, so the combined outgoing flow of v_i is strictly less than d_i. By flow conservation, the incoming flow to v_i must also be strictly less than d_i, implying $v_i \notin S$ as claimed.