Introduction to Algorithms **Solution Sketches for Problem Set 8**
CS 4820, Spring 2018

(1) *(8 points) At Cornell University, doctoral students in Computer Science can earn their Ph.D. either in Ithaca or at the new Cornell Tech campus in New York City. However all students initiate their study in Ithaca and complete some course requirements before potentially moving to New York City. This inspires the following course scheduling problem.*

A department has p professors and $2p$ courses. Each professor teaches two courses, one per semester. Assume that professors are numbered $0, \ldots, p-1$ and courses are numbered $0, \ldots, 2p-1$ such that professor i teaches courses $2i$ and $2i+1$, for each $i = 0, \ldots, p-1$. There are s students (numbered $0, \ldots, s-1$) each of whom is trying to finish their remaining course requirements in the fall semester so that they can move to a different campus in the spring. The remaining course requirements for student j are fulfilled by taking at least $n(j)$ courses from a specified list $L(j)$.

The Impatient Grad Students Problem (IGSP) *is the following decision problem: given the numbers $p, s, n(0), \ldots, n(s-1)$ (in binary) and the lists $L(0), \ldots, L(s-1)$, decide whether the courses can partitioned into the fall and spring semesters such that*

1. *each professor teaches one fall course and one spring course*
2. *for each list $L(j)$, at least $n(j)$ of the courses in $L(j)$ are scheduled in the fall semester.*

Prove that IGSP *is NP-complete.*

Solution. Proving that the problem is in NP is slightly subtle, because the integer p is given in binary, so the obvious approach of specifying a partition of $\{0, 1, \ldots, 2p-1\}$ doesn't work. (The size of that set might be exponential in the input size.) Instead, the verifier takes an input instance, and a partition of the set $L = \bigcup_{j=0}^{s-1} L(j)$ into two sets F (fall) and S (spring), and it checks that:

1. Each of the lists $L(j)$ $(0 \leq j \leq s-1)$ has at least $n(j)$ elements in F.
2. For each even number $2i \in F$, we have $2i+1 \notin F$.
3. For each even number $2i \in S$, we have $2i+1 \notin S$.

This works because if the problem is a 'yes' instance of IGSP then there exists a partition of $\{0, 1, \ldots, 2p-1\}$ that satisfies the problem constraints, and the restriction of this partition to L will pass all of the verifier's tests. Conversely, if there is a partition of L that passes all of the verifier's tests, this can be extended to a partition of $\{0, 1, \ldots, 2p-1\}$ that satisfies all of the problem constraints as follows. For every professor whose two courses both belong to L, those two courses have already been split between the fall and spring semesters. For every professor who teaches one course in L and one course in its complement, assign the extra course to the semester when the professor does not yet have a teaching assignment. Finally, for a professor neither of whose courses belongs to L, split them arbitrarily between the fall and spring semesters.

To prove that the problem is NP-complete, we reduce from 3SAT. Each variable corresponds to a professor. Each clause corresponds to a student j with $n(j) = 1$ and $L(j)$ equal to a list of courses corresponding to the literals of the clause. In particular, if clause j contains literal x_i then $L(j)$ contains $2i$ and if clause j contains literal $\bar{x}_i$ then $L(j)$ contains $2i+1$. The running time of the reduction is $O(n+m)$ (number of variables plus clauses) which is the time required to build all of the lists.

Given a satisfying truth assignment, we can partition the courses by scheduling $2i$ in the fall and $2i+1$ in the spring if $x_i =$ TRUE and scheduling $2i+1$ in the fall and $2i$ in the spring if $x_i =$ FALSE. By construction each professor teaches one fall course and one spring course. The fact that each clause contains at least one satisfied literal implies the second property, that for each list $L(j)$ there are at least $n(j)$ courses in $L(j)$ that are scheduled for the fall semester.

(2) *(10 = 2+8 points) In the* PAIRED INTERVAL SCHEDULING *problem one is given sets* $S_1, \ldots, S_n \subset \mathbb{R}$*, each of which is a union of two disjoint closed intervals with distinct integer endpoints. In other words,* $S_i = [a_i, b_i] \cup [c_i, d_i]$ *where* $a_i < b_i < c_i < d_i$ *are all integers. One is also given a positive integer* $k \leq n$*. The problem is to decide whether there exist* k *pairwise disjoint sets among the collection* $\{S_1, \ldots, S_n\}$*.*

(a) (2 points) *What is wrong with the following incorrect proof that* PAIRED INTERVAL SCHEDULING *is NP-complete?*

> *Faced with an instance of the* PAIRED INTERVAL SCHEDULING *problem, we can form a* conflict graph *whose vertex set is* $\{1, \ldots, n\}$*, with an edge joining* i *to* j *if* S_i *and* S_j *intersect. The problem asks whether this graph contains an independent set of size* k*. The* INDEPENDENT SET *problem is NP-Complete, so the* PAIRED INTERVAL SCHEDULING *problem is also NP-Complete.*

(b) (8 points) *Prove that* PAIRED INTERVAL SCHEDULING *is NP-complete.*

Solution. For part (a), the fallacy is that the paragraph describes a reduction from PAIRED INTERVAL SCHEDULING to INDEPENDENT SET. However, this merely shows that PAIRED INTERVAL SCHEDULING belongs to NP. To show that it is NP-complete, we would also need to reduce from a known NP-complete problem (such as INDEPENDENT SET) to PAIRED INTERVAL SCHEDULING. In other words, the representation of PAIRED INTERVAL SCHEDULING using a conflict graph shows that it is equivalent to a *special case* of INDEPENDENT SET. However, to show PAIRED INTERVAL SCHEDULING is NP-complete, we need to show that the *general case* of INDEPENDENT SET reduces to it.

For part (b), the paragraph quoted in part (a) specifies a reduction to INDEPENDENT SET, which suffices to show that PAIRED INTERVAL SCHEDULING is in NP. Alternatively, and more directly, there is a verifier that takes an instance of PAIRED INTERVAL SCHEDULING, and a set of indices $\{i(1), \ldots, i(k)\}$, checks that the elements of this index set are pairwise distinct, and checks that for all $q \neq r$ the sets $S_{i(q)}$ and $S_{i(r)}$ are disjoint. It only takes constant time to perform this disjointness test for a given pair $S_{i(q)}$ and $S_{i(r)}$ because checking whether a given two intervals intersect only takes constant time, and each of the sets $S_{i(q)}$, $S_{i(r)}$ is made up of two disjoint intervals, hence we only need to perform four such interval-intersection tests to see whether $S_{i(q)}$ and $S_{i(r)}$ are disjoint.

To prove that PAIRED INTERVAL SCHEDULING is NP-complete we reduce from 3SAT. Given an instance of 3SAT with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$, we create an instance of PAIRED INTERVAL SCHEDULING with the following structure.

- For each variable x_i we create two interval pairs:

$$T_i = [2i-1, 2i] \cup [-4mi, -4mi + 2m - 1], \quad F_i = [2i-1, 2i] \cup [-4mi + 2m, -4mi + 4m - 1].$$

- For each clause C_j we create three interval pairs

$$S_j = [2n+2j-1, 2n+2j] \cup X_j, \quad T_j = [2n+2j-1, 2n+2j] \cup Y_j, \quad U_j = [2n+2j-1, 2n+2j] \cup Z_j$$

  where X_j, Y_j, Z_j are unit-length intervals corresponding to the three literals in C_j. If the literal x_i belongs to C_j then the corresponding unit-length interval is a subinterval of $[-4mi + 2m, -4mi + 4m - 1]$; if $\bar{x}_i$ belongs to C_j then the corresponding unit-length interval is a subinterval of $[-4mi, -4mi + 2m - 1]$. In both cases, the unit-length interval corresponding to the literal begins $2j - 2$ units to the right of the left endpoint of the length $2m - 1$ interval in which it is embedded.

Finally, we define the desired number of interval-pairs, k, to be equal to $n + m$.

The reduction creates $2n + 3m$ interval pairs, and it takes $O(1)$ to construct each one from the input data, so the reduction runs in linear time.

To prove correctness, we must show two things.

If the 3-SAT instance has a satisfying truth assignment, our reduction creates an instance of PAIRED INTERVAL SCHEDULING **in which there are $n+m$ disjoint sets.** Given a satisfying truth assignment, let Ω denote a collection of sets chosen as follows: for each variable x_i, select $T_i \in \Omega$ if the truth assignment sets $x_i =$ TRUE and $F_i \in \Omega$ if the truth assignment sets $x_i =$ FALSE. For each clause C_j, by assumption there is at least one literal in C_j that is satisfied by the given truth assignment. Take the set $S_j, T_j,$ or U_j corresponding to this literal, and include it in Ω. By construction, Ω clearly has $n + m$ elements. They are pairwise disjoint because no two of the selected "literal sets" overlap (they correspond to different variables, and the reduction was constructed so that sets belonging to distinct "variable gadgets" cannot overlap), no two of the selected "clause sets" overlap (same reasoning: they correspond to different clauses, and the reduction was constructed so that sets belonging to distinct "clause gadgets" cannot overlap), and none of the selected clause sets overlaps the selected variable sets because our construction ensures that the unit-length interval X_j, Y_j, or Z_j corresponding to the satisfied literal in C_j belongs to the "long interval" of a variable-gadget set that was *not* selected to belong to Ω.

If the reduction creates an instance of PAIRED INTERVAL SCHEDULING **in which there are $n + m$ disjoint sets, then the original** 3SAT **instance has a satisfying truth assignment.** Note that the sets T_i, F_i intersect one another, and that the sets S_j, T_j, U_j all intersect one another. Hence, a collection Ω of $n + m$ disjoint sets among those constructed by our reduction must consist of *exactly one set* from each of the collections $\{T_i, F_i\}_{i=1}^{n}$ and $\{S_j, T_j, U_j\}_{j=1}^{m}$. Now construct a truth assignment as follows: if $T_i \in \Omega$ then set $x_i =$ TRUE, and if $F_i in \Omega$ then set $x_i =$ FALSE. As noted earlier, our assumption that Ω consists of $n + m$ pairwise disjoint sets implies that exactly one of T_i, F_i belongs to Ω so this definition of the truth assignment is unambiguous. Finally, to confirm that the constructed truth assignment is a satisfying assignment, observe that for each clause there is a set S_j, T_j, or U_j that belongs to Ω; this set identifies a literal of clause j, and the literal must be satisfied by the chosen truth assignment as the sets corresponding to that literal would already belong to Ω, and would overlap with the set in Ω that corresponds to clause j.

Alternate solution. There is an alternative reduction from INDEPENDENT SET that is much more complicated. We present this alternative reduction here, since it may be instructive. Given an instance (G, k) of INDEPENDENT SET, we name the vertices $v_1, \ldots, v_n$ and the edges $e_1, \ldots, e_m$. We then construct an instance of PAIRED INTERVAL SCHEDULING defined as follows. The definition specifies a set of $3m + (8m + n) + k$ unit-length intervals, and each of the sets in our PAIRED INTERVAL SCHEDULING problem will be a union of two of the specified intervals.

- For $j = 1, \ldots, m$ there are 3 intervals devoted to the gadget representing edge e_j. Letting u, v denote the endpoints of e_j, the three intervals are

$$X_{j,u} = [6j - 5, 6j - 4], \quad X_{j,0} = [6j - 3, 6j - 2], \quad X_{j,v} = [6j - 1, 6j].$$

- For $i = 1, \ldots, n$, if vertex v_i has degree d_i then there are $4d_i + 1$ intervals devoted to the gadget representing vertex v_i. These are numbered $Y_{i,p}$ for $p = 0, 1, \ldots, 4d_i$.

$$Y_{i,p} = [s_i + p, s_i + p + 1], \quad \text{where } s_i = 6m + 1 + \sum_{j<i} (4d_j + 2).$$

- For $\ell = 1, \ldots, k$ let
$$Z_\ell = [-2\ell, -2\ell + 1].$$

Observe that all of the intervals listed above are pairwise disjoint except for the pairs $Y_{i,p-1}$ and $Y_{i,p}$, for $1 \leq i \leq n$ and $1 \leq p \leq 2d_i$. We now proceed to define an instance of Paired Interval Scheduling by specifying a collection of sets, each of which is a union of two disjoint intervals chosn from the collection identified above.

- For each edge e_j we have two sets
$$F_{j,u} = X_{j,u} \cup X_{j,0}, \quad F_{j,v} = X_{j,v} \cup X_{j,0}.$$
- For each vertex v_i we number that edges that contains v_i from 0 to $d_i - 1$. For each $q = 0, \ldots, d_i - 1$, if e_j is the edge that occurs in the q^{th} position on this list, we have sets
$$G_{i,q} = Y_{i,2q} \cup X_{j,v}, \quad H_{i,q} = Y_{i,2q+1} \cup Y_{i,2q+2d_i+2}.$$
- For each vertex v_i and for $\ell = 1, 2, \ldots, k$ we have a set
$$I_{i,\ell} = Y_{i,2d_i} \cup Z_\ell.$$

Finally, it sets the target number of sets equal to $3m + k$.

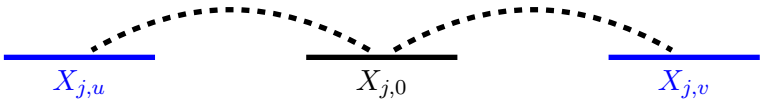

Figure 1: Edge gadget

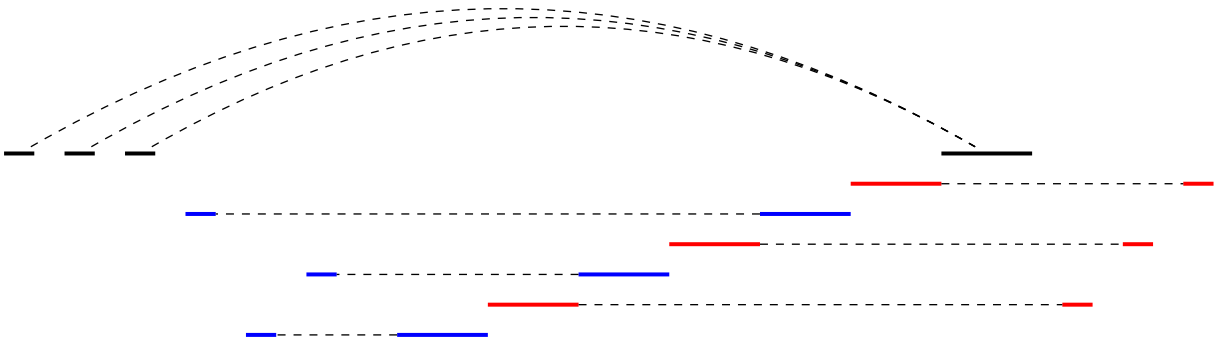

Figure 2: Vertex gadget

This completes the description of the reduction. After spending $O(m + n + k)$ time precomputing the endpoints of intervals X_*, Y_*, Z_*, the reduction creates $O(m + nk)$ sets, taking constant time to create each one, so the total running time of the reduction is $O(m + nk)$.

To prove correctness of the reduction, we first describe how to transform an independent set S of size k in G into a collection of $3m + k$ pairwise disjoint sets. For each edge e_j, let u denote an endpoint of e_j that does not belong to S, and include $F_{j,u}$ in our collection. For each vertex v_i, if v_i belongs to S then include the sets $G_{i,q}\,(0 \leq q < d_i)$ in our collection, and otherwise include the sets $H_{i,q}\,(0 \leq < d_i)$ in our

collection. Finally, let ϕ be any bijection between S and $\{1, \ldots, k\}$, and for each $v_i \in S$ let $\ell = \phi(v_i)$ and include the sets $I_{i,\ell}$ in our collection. The number of sets in the collection is

$$m + \left(\sum_i d_i \right) + k = 3m + k$$

as desired, and they are pairwise disjoint.

Conversely, if we are given a collection $\mathcal{C}$ of $3m+k$ pairwise disjoint sets from among the sets constructed by our reduction, we let S denote the set of vertices v_i of our original graph such that $\mathcal{C}$ contains $I_{i,\ell}$ for some ℓ. We claim S is an independent set of size k. To prove it, note that any collection of paiwise disjoint sets from among the sets constructed by our reduction includes:

- for each j, at most one of the sets $F_{j,u}$ and $F_{j,v}$ since they have $X_{j,0}$ in common.
- for each i, q, at most one of the sets $G_{i,q}$ and $H_{i,q}$ since they have the point $s_i + 2q + 1$ in common.
- for each ℓ, at most one of the sets $I_{i,\ell}\,(i = 1, \ldots, n)$ since they all have Z_ℓ in common.

The total number of sets therefore can be at most $m + (\sum_i d_i) + k$, which (as noted above) is equal to $3m + k$. Furthermore, the only way to have $3m + k$ pairwise disjoint sets is if we include exactly one set from each of the groups enumerated in the bulleted list above. In particular, this means that we must have k sets of the form $I_{i,\ell}$. These sets must correspond to distinct vertices because if $I_{i,\ell}$ and $I_{i,\ell'}$ were both included, they would overlap on the interval $Y_{i,2d_i}$ contradicting their presumed disjointness. Hence, the set S consists of k distinct vertices of G. It remains to check that it is an independent set.

If v_i belongs to S, it means that $I_{i,\ell}$ is in our collection of pairwise disjoint sets, for some ℓ. This means that H_{i,d_i-1} is not in our collection, since H_{i,d_i-1} and $I_{i,\ell}$ intersect. Because one of G_{i,d_i-1} or H_{i,d_i-1} must be included, this means G_{i,d_i-1} is in our collection. By downward induction on q, none of the sets $H_{i,q}$ and all of the sets $G_{i,q}$ belong to our collection.

If e_j is an edge with endpoints $u = v_i$ and $v = v_{i'}$, then one of the sets $F_{j,u}$, $F_{j,v}$ belongs to our collection; suppose without loss of generality that it is $F_{j,u}$. Since the interval $X_{j,u}$ belongs to $F_{j,u}$, it means that our collection cannot include the other set that contains $X_{j,u}$, namely $G_{i,q}$ where q is the index such that e_j is the q^{th} element of the list of edges of vertex $u = v_i$. Since the preceding paragraph proved that $G_{i,q}$ is included in our collection whenever $v_i \in S$, it follows that $v_i \notin S$. We have thus shown that every edge e_j has an endpoint that does not belong to S, which implies that S is an independent set.

(3) *(12 = 2+10 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 problem asks you to prove that* Flow Saturation *is NP-complete.*

(a) *(2 points) What is wrong with the following incorrect solution?*

> The Flow Saturation problem is in NP because it has a polynomial-time verifier that takes the pair (G, k) along with a flow f, and checks that f satisfies conservation and capacity constraints (in linear time). While checking capacity constraints it also keeps a counter of how many edges are saturated, and it reports "yes" if the conservation and capacity constraints are satisfied, and the number of saturated edges is at least k.
>
> To prove that Flow Saturation is NP-complete we reduce from Hamiltonian Path. Given a directed graph G_0 that is an instance of Hamiltonian Path, our reduction creates

a new graph G consisting of a copy of G_0 together with four extra vertices $\{s, s', t', t\}$. G has two new edges (s, s') and (t', t), and it also has edges from s' to every vertex of G_0 and from every vertex of G_0 to t'. Finally we set all edge capacities to 1, and we treat this as an instance of FLOW SATURATION with source s, sink t, and parameter $k = n + 3$. The reduction takes linear time: if G_0 has n vertices and m edges, then G has $n+4$ vertices and $m+2n+2$ edges, and it takes constant time to insert each vertex or edge into the adjacency list representation of G.

To prove the correctness of the reduction, we show the following two statements.

If G_0 has a Hamiltonian path, then G has a flow that saturates $n + 3$ edges. Indeed, if the Hamiltonian path P_0 in G_0 is from s_0 to t_0, then G contains a path P of length $n+3$ from s to t, that begins with s, s', s_0, then traverses the entire path P_0 to reach t_0, then ends with t_0, t', t. Sending one unit of flow on this path P saturates all of its $n + 3$ edges.

If G has a flow that saturates $n + 3$ edges, then G_0 has a Hamiltonian path. Indeed, since G has just a single unit-capacity edge leaving s, the max-flow value in G is equal to 1 and so a maximum flow consists of a single path from s to t. If this path saturates $n + 3$ edges, then two of its edges leave s and s', respectively, two of its edges come into t' and t, respectively, and the remaining $n - 1$ edges belong to G_0. Those $n - 1$ edges must form a Hamiltonian path in G_0.

(b) *(10 points) Prove that the* FLOW SATURATION *problem is indeed NP-complete.*

REMARK: Part (a) contains a valid proof that FLOW SATURATION **belongs to NP. Therefore, in doing part (b), you do not need to include that step in your solution.**

Solution. For part (a), the bug is that in the flow network created by the reduction, a maximum flow consists of a single path from s to t **plus any number of edge-disjoint cycles**. Hence a maximum flow might saturate $n + 3$ edges even though G_0 does not have a Hamiltonian path. The simplest case in which this happens is when G_0 is a disconnected graph with $n = 5$ vertices forming two connected components: an edge with endpoints u, v and a 3-cycle with vertices a, b, c. Then in G there is a flow that sends one unit on the path $s \to s' \to u \to v \to t' \to t$ and one unit on the cycle $a \to b \to c \to a$. This flow saturates $n + 3 = 8$ edges, but it does not imply that G_0 is Hamiltonian.

Given an instance (G, k) of INDEPENDENT SET, our reduction creates an instance of FLOW SATURATION with the following nodes and edges.

- source s, sink t
- node x_e for each edge e of G
- node y_v for each vertex v of G
- edges (s, x_e) with capacity 1 for all edges e
- edge (x_e, y_v) with capacity 1 whenever e is an edge of G and v is one of its endpoints.
- edge (y_v, t) with capacity $d(v)$, the degree of v in G, for each vertex v of G.

The target value (number of edges to be saturated) in this FLOW SATURATION problem is $2m + k$.

The reduction runs in polynomial time because it constructs a flow network with $2 + n + m$ nodes and $3m + n$ edges, and creating each of them takes only $O(1)$ time. To prove correctness of the reduction we need to establish two facts.

If G has an independent set of size k, then our reduction creates a flow network having a flow that saturates at least $2m + k$ edges. Letting S denote the independent set of size k, we define a flow f as follows. First, $f(s, x_e) = 1$ for all e. Second, if e is an edge with endpoints u, v, then we let v denote the endpoint which belongs to S (or an arbitrary endpoint, if neither endpoint belongs to S) and we set $f(x_e, y_v) = 1$, $f(x_e, y_u) = 0$. Finally we choose the flow values $f(y_v, t)$ so as to satisfy flow conservation. This flow saturates one incoming edge and one outgoing edge of each node x_e — $2m$ edges in total — and it also saturates the outgoing edges of y_v for all $v \in S$, which amounts to another k edges.

If the flow network created by our reduction has a flow saturating at least $2m + k$ edges, then G has an independent set of size k. Let S denote the set of vertices v such that the flow saturates edge (y_v, t). The capacity of (y_v, t) equals the combined capacity of the incoming edges of y_v, hence each of those edges must also be saturated. For each edge $e = (u, v)$ at most one of the edges (x_e, y_u), (x_e, y_v) can be saturated, since both of those edges have capacity 1 and the only incoming edge of x_e also has capacity 1. Hence, it cannot be the case that both endpoints of an edge, u and v, belong to S. In other words, S is an independent set. Since each node x_e has at most one saturated incoming edge and at most one saturated outgoing edge, $2m$ edges in total, it follows that at least k of the remaining edges — which are all of the form (y_v, t) — must be saturated. Hence S is an independent set of size at least k. If the size of S is not exactly k, then any k-element subset of S is an independent set of size exactly k.