Solutions to Quiz 4

Solution 1:

(a) (i) $O(n^2 + m)$: Strong polynomial time

(ii) $O((n + m) \log n)$: Strong polynomial time

(iii) $O(n^{\log m})$: Neither (since the exponent is not a constant)

(iv) $O(n + m \cdot W)$: Neither (since the input size of weights is proportional to $\lg W$, this is exponentially larger)

(v) $O(n + m \cdot \log W)$: Weak polynomial time

(b) (i) P $\subseteq$ NP: True (If a problem is solvable in polynomial time, it is verifiable in polynomial time.)

(ii) NP $\subseteq$ P: Unknown to science (Since it is possible that P = NP.)

(iii) If $A \in$ P and $B \in$ NP-complete then $A \leq_P B$: True (If B is NP-complete, then by definition it is NP-hard. Since P $\subseteq$ NP, $A \in$ NP. By definition, all problems in NP are polynomially reducible to any NP-hard problem, so $A \leq_P B$.)

(iv) If $A \leq_P B$ and $A \notin$ P then $B \notin$ P: True ($A \leq_P B$ means that we can use a subroutine for B to solve A in polynomial time. Therefore, if $A \notin$ P, there cannot be a polynomial time solution to B.)

(v) Determining whether a graph is connected is NP-complete: Unknown to science (Determining whether a graph is connected is certainly in P (by DFS), but if P = NP, then all NP-complete problems are in P and vice versa.)

Solution 2: (This is the same as Problem 2 on Practice Problems 9, but with multiple sources.) To solve the separation problem, create an s-t network as follows. Starting with the directed graph G with start vertices S and terminal vertices T, create a super-source node s^* and super-sink node t^*. Add edges (s^*, s_i), for all $s_i \in S$, and add edges (t_j, t^*), for all $t_j \in T$ (see Fig. 1). Set all the edge capacities of the original edges to 1. Set the capacities of the edges (s^*, s_i) and (t_j, t^*) to $+\infty$. (Generally, the capacities on these edges needs to be large enough to allow for all the flow through each of these vertices.) Let G' denote the resulting network.

Next, compute a min-cut in G'. This can be done by computing a max-flow f in G', finding all the vertices X that are reachable from s^* in the residual network G'_f and letting Y denote the remaining edges. By the Min-cut/Max-flow Theorem and the fact that all edge capacities were set to 1, the number of edges of G that cross the cut (going from X to Y) will equal the max-flow value, which equals the number of edges in the min-cut. Therefore, this set of edges constitute the smallest subsetr of edges that separate all the start vertices from the terminal vertices.

The running time is just the time needed to compute G' (which is $O(n + m)$) and the time for the network flow algorithm.

Solution 3: (See the solution to problem 3(b) on Practice Problems 11.) Note that it is important that when an edge (or other gadgetry) is added to fix the color of some vertex, these edges need to

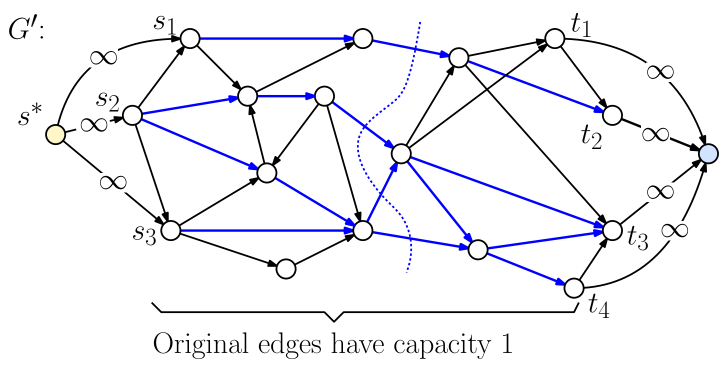

Figure 1: Eliminating edges to separated r from terminals. Saturated edges in the max flow are shown in blue and the min-cut is shown with the dashed curve.

remain in the graph. The reason is that there may be multiple 3-colorings, and if the edges/gadgets are removed, the vertices may be colored in an inconsistent manner.

Solution 4: (This is the same as Problem 5 on Practice Problems 11, but with degree requirement of $2k$.) We show that IS $\leq_P$ UDIS, where IS is the standard independent set problem. Consider an input (G, k) to the independent set problem. Since we do not know which vertices of G are in the independent set (or even whether an independent set exists), our approach will be to add a bunch of bogus vertices whose job it is to make the degree of *every* vertex at least as high as $2k$, but to do so in such a way that we neither create nor destroy any existing independent sets.

First make a copy of G and add $2k$ new vertices to it. Add edges between all these new vertices. Also, for each original vertex in G, add add $2k$ edges joining it to each of the new vertices. Let G' denote the resulting graph. Output (G', k) (see Fig. 2). Clearly, this can be done in polynomial time, and in fact in time $O(n + m + kn)$.

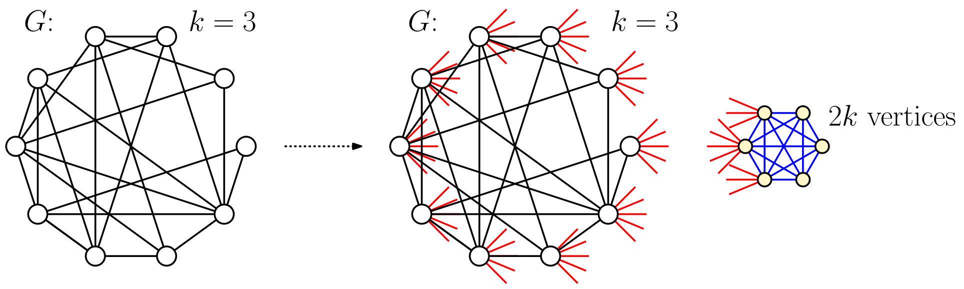

Figure 2: Reducing IS to UDIS. Red edges connect every vertex of the original graph to the clique of size $2k$.

Notice that G' has $n' = n + k$ vertices, and each vertex of G' has degree at least k (because it is adjacent to all the newly added vertices in the clique induced by the new vertices). It was not necessary to prove correctness, but for completeness, here is a proof.

Lemma: G has an independent set of size k if and only if G' has a UDIS of size k.

Proof:
- ($\Rightarrow$) If V' is an IS in G of size k, then the corresponding vertices of G' are not adjacent

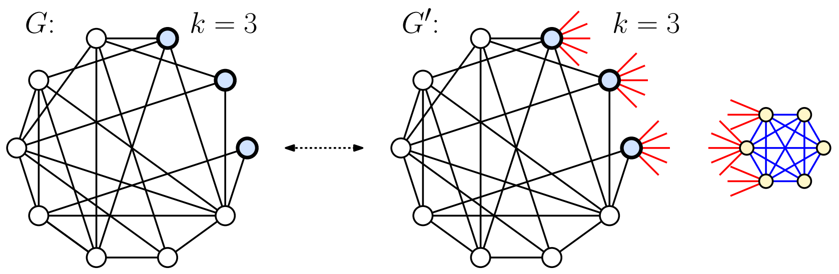

Figure 3: Correctness of the reduction.

to each other, but they all have degree at least $2k$ (since every vertex in G does). Thus, V' is a UDIS in G' of size k (see Fig. 3).

- ($\Leftarrow$) Suppose that V' is a UDIS in G' of size k. We may assume that $k \geq 2$, since otherwise the problem is trivial. We claim that there can be no vertices in G' from the clique of new vertices (excluding the trivial case where $k = 1$). The reason is that these vertices are adjacent to each other and every other vertex in G'. Therefore the vertices of V' come from the original vertices of G, implying that V' forms an independent set in G of size k.

Solution 5:

(a) We solve the ball redistribution problem via reduction to circulations with vertex demands and lower and upper flow capacities on each edge. We will create a bipartite graph, where the supply values on the left side reflect the original number of balls and demand values on the right reflect the desired final number of balls.

We generate a network $G = (V, E)$ as follows. We create two sets of vertices, one for the initial buckets, denoted $\{u_1, \ldots, u_n\}$, and one for the final buckets, denoted $\{v_1, \ldots, v_n\}$. There are two solutions, one based on using vertex demands to control the flow through each vertex and the other based on edge capacities to control the flow. We will describe the former (see Fig. 4(a)).

- For $i \in [1, n]$, and $j \in [1, n]$, we set $d[u_i] = -a_i$, which models the fact that bucket i starts with a_i balls. We set $d[v_j] = b_j$, which models the fact that bucket j must receive b_j balls. Note that flow along the edge (u_i, v_i) reflects the number of balls that remain in bucket i.
- For $i \in [1, n]$, and $j \in [1, n]$, we add edge (u_i, v_j) if $|i - j| \leq 2$, that is, if bucket j is within 2 buckets of bucket i. If $i = j$ (same bucket), we set the capacity range to $[1, \infty]$, which models the fact that at least one ball must remain in its same bucket. (Since the flow through this edge cannot exceed a_i or b_j, we could also use these values as capacity upper bounds.) If $i \neq j$, we set the capacity range to $[0, 2]$, indicating that at most 2 balls can move from any bucket to any different bucket.

The alternative is to set all vertex capacities to zero and generate a source vertex s and sink vertex t, and generate edges (s, u_i), each with capacity range $[a_i, a_i]$ and the edges (v_i, t), each with capacity $[b_i, b_i]$, and create an edge (t, s) with capacity $[0, \infty]$ (see Fig. 4(b)). Yet another variant is to remove the edge from t to s and set $d[s] = -m$ and $d[t] = m$ (where m is the total number of balls).

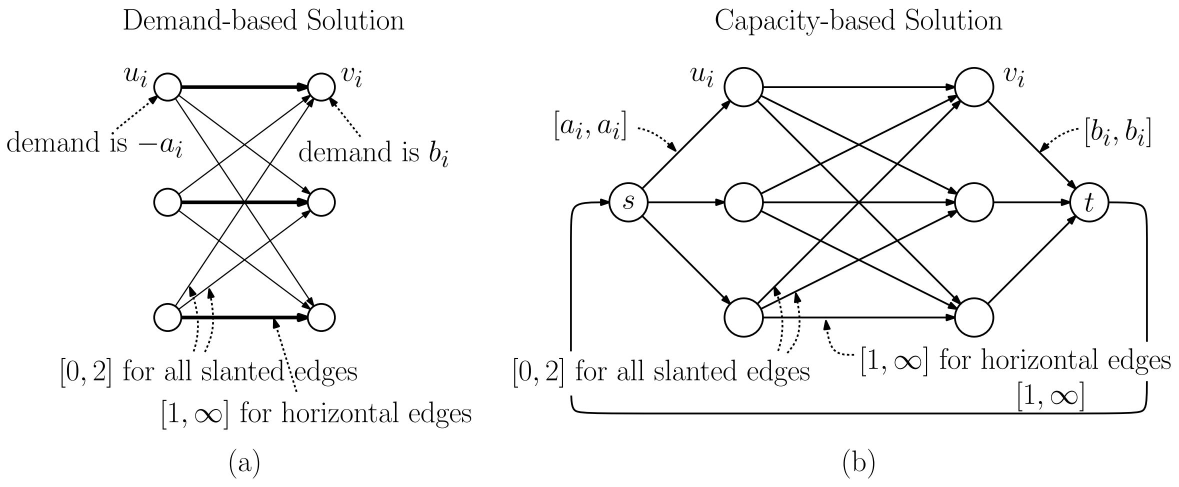

Figure 4: Reducing the bucket redistribution problem to circulation.

(b) An example for the input $A = \langle 5, 3, 1 \rangle$ and $B = \langle 2, 2, 5 \rangle$ is shown in Fig. 5(a) for the demand-based solution Fig. 5(a) for the capacity-based solution.

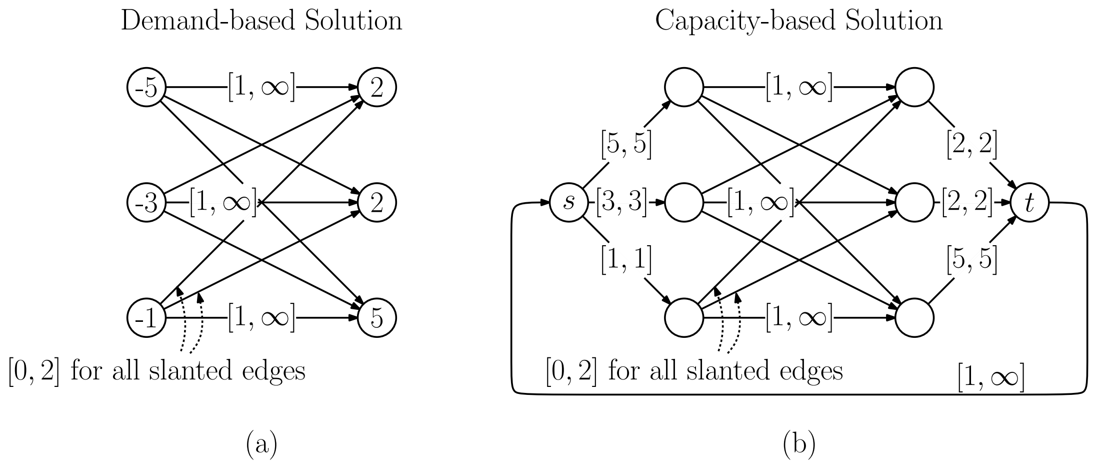

Figure 5: Example of the reduction.