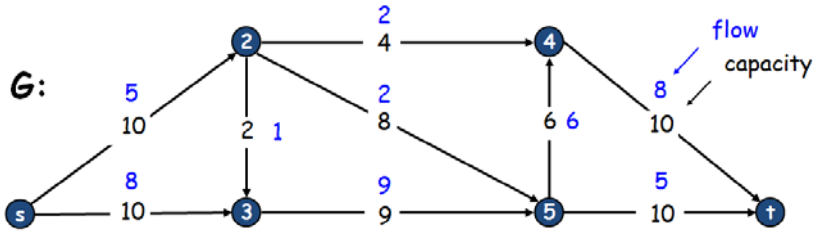
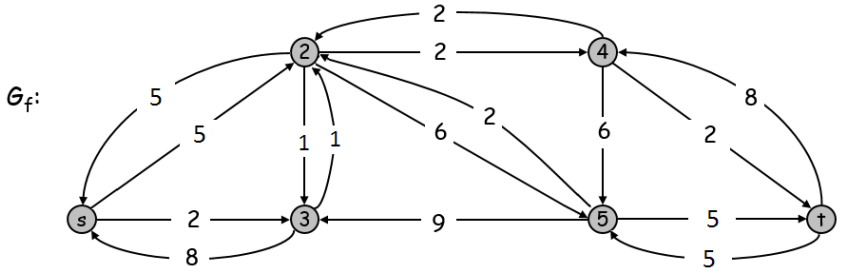
COT5405: Homework 4 (Fall 2017)

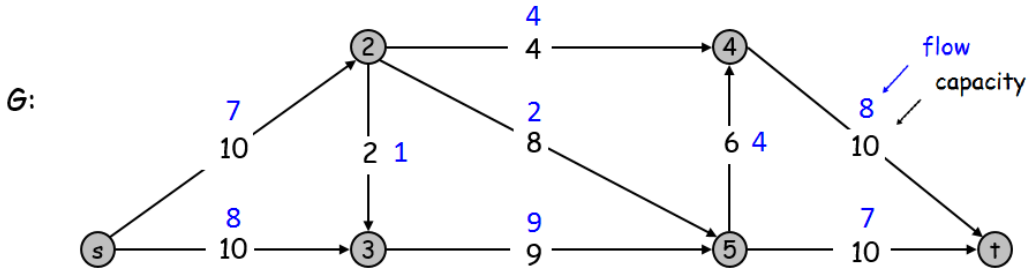
1. **Ford-Fulkerson Algorithm**: Study the example on slides 'ch7-maxflow.ppt'. The same graph is used for this question. Suppose the flow values (blue numbers) on all edges at current step in running the Ford-Fulkerson algorithm.



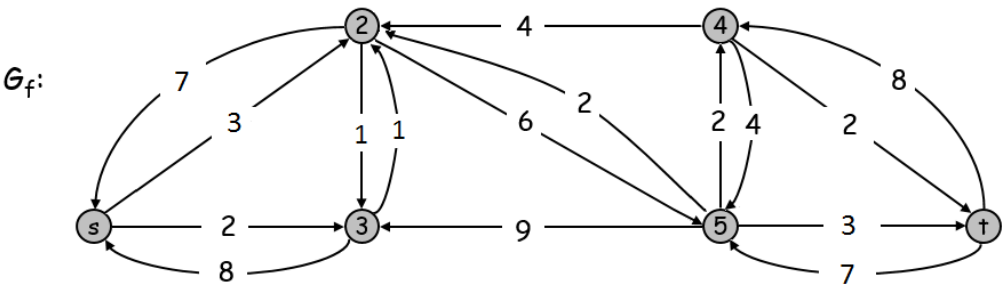
a. The corresponding Residual graph $\mathbf{G_f}$, the overall flow value of which is 13:



b. For the augmenting path s→2→4→5→t, the bottleneck equals 2. Thus we can augment the flow along this path by 2. New overall flow value is 15. Updated graph G is:



c. The new updated corresponding Residual graph $\mathbf{G_f}$



2. Exercise 8 on page 418.

To solve this problem, construct a graph $G = (V, E)$ as follows: Let the set of vertices consist of a super-source node, four supply nodes (one for each blood type) adjacent to the source, four demand nodes and a super sink node that is adjacent to the demand nodes. For each supply node u and demand node v, construct an edge (u, v) if type v can receive blood from type u and set the capacity to ∞ or the demand for type v. Construct an edge (s, u) between the source s and each supply node u with the capacity set to the available supply of type u. Similarly, for each demand node v and the sink t, construct an edge (v, t) with the capacity set to the demand for type v.

Now compute an (integer-valued) maximum flow on this graph. Since the graph has constant size, the scaling max-flow algorithm takes time $O(\log C)$, where C is the total supply, and the Preflow-Push algorithm takes constant time.

We claim that there is sufficient supply for the projected need. if and only if the edges from the demand nodes to the sink are all saturated in the resulting max-flow. Indeed, if there is sufficient supply, in which s_{ST} of type S are used for type T, then we can send a flow of s_{ST} from the supply node of type S to the demand node of type T, and respect all capacity conditions. Conversely, if there is a flow saturating all edges from demand nodes to the sink, then there is an integer flow with this property; if it sends f_{ST} units of flow from the supply node for type S to the demand node for type T, then we can use f_{ST} nodes of type S for patients of type T.

(b) Consider a cut containing the source, and the supply and demand nodes for B and A. The capacity of this cut is $50 + 36 + 10 + 3 = 99$, and hence all 100 units of demand cannot be satisfied.

An explanation for the clinic administrators: There are 87 people with demand for blood types O and A; these can only be satisfied by donors with blood types O and A; and there are only 86 such donors.

Note. We observe that part (a) can also be solved by a greedy algorithm; basically, it works as follows. The O group can only receive blood from O donors; so if the O group is not satisfied, there is no solution. Otherwise, satisfy the A and B groups using the leftovers from the O group; if this is not possible, there is no solution. Finally, satisfy the AB group using any remaining leftovers. A short explanation of correctness (basically following the above reasoning) is necessary for this algorithm, as it was with the flow algorithm.

3. Exercise 9 on page 419.

We build the following flow network. There is a node v_i for each patient i, a node w_j for each hospital j, and an edge (v_i, w_j) of capacity 1 if patient i is within a half hour drive of hospital j. We then connect a super-source s to each of the patient nodes by an edge of capacity 1, and we connect each of the hospital nodes to a super-sink t by an edge of capacity $\lceil n/k \rceil$.

We claim that there is a feasible way to send all patients to hospitals if and only if there is an s-t flow of value n. If there is a feasible way to send patients, then we send one unit of flow from s to t along each of the paths s, v_i, w_j, t, where patient i is sent to hospital j. This does not violate the capacity conditions, in particular on the edges (w_j, t), due to the load constraints. Conversely, if there is a flow of value n, then there is one with integer values. We send patient i to hospital j if the edge (v_i, w_j) carries one unit of flow, and we observe that the capacity condition ensures that no hospital is overloaded.

The running is the time required to solve a max-flow problem on a graph with $O(n + k)$ nodes and $O(nk)$ edges.

4. Exercise 1, page 505

(1a) Yes. One solution would be: *Interval Scheduling* can be solved in polynomial time, and so it can also be solved in polynomial time with access to a black box for *Vertex Cover*. (It need never call the black box.) Another solution would be: *Interval Scheduling* is in NP, and anything in NP can be reduced to *Vertex Cover*. A third solution would be: we've seen in the book the reductions *Interval Scheduling* $\leq_P$ *Independent Set* and *Independent Set* $\leq_P$ *Vertex Cover*, so the result follows by transitivity.

(1b) This is equivalent to whether P = NP. If P = NP, then *Independent Set* can be solved in polynomial time, and so *Independent Set* $\leq_P$ *Interval Scheduling*. Conversely, if *Independent Set* $\leq_P$ *Interval Scheduling*, then since *Interval Scheduling* can be solved in polynomial time, so could *Independent Set*. But *Independent Set* is NP-complete, so solving it in polynomial time would imply P = NP.

5. Explain why if $x_0 \Leftrightarrow x_1 \wedge x_2 \Rightarrow$ add 3 clauses to the CNF: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$
(see slide 19, NP-Complete.pptx)

Negation symbol: ¬x same as $\overline{x}$

$x_0 \Leftrightarrow x_1 \wedge x_2$: $(x_0 \Rightarrow x_1 \wedge x_2)$ $\wedge (x_1 \wedge x_2 \Rightarrow x_0)$

: $[(x_0 \Rightarrow x_1) \wedge (x_0 \Rightarrow x_2)$ $\wedge (x_1 \wedge x_2 \Rightarrow x_0)]$

: $[(\neg x_0 \vee x_1) \wedge (\neg x_0 \vee x_2)$ $\wedge (\neg x_0 \Rightarrow \neg (x_1 \wedge x_2)]$

: $[(\neg x_0 \vee x_1) \wedge (\neg x_0 \vee x_2)$ $\wedge (\neg x_0 \Rightarrow (\neg x_1 \vee \neg x_2))]$

: $[(\neg x_0 \vee x_1) \wedge (\neg x_0 \vee x_2)$ $\wedge (x_0 \vee \neg x_1 \vee \neg x_2)]$