CS218 Design and analysis of algorithms **Jan-Apr 2024**

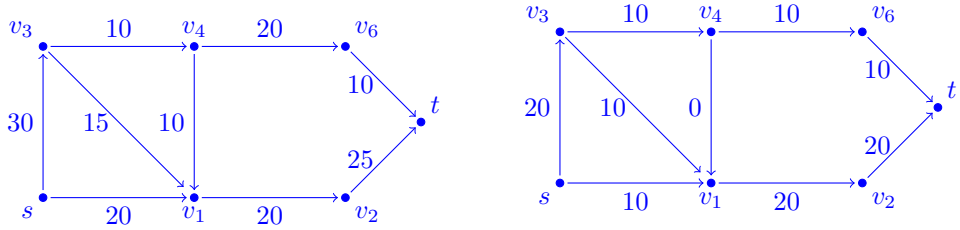
Quiz 2 solutions

Total Marks: 20 *Time: 55 minutes*

Instructions.

- Please write your answers concisely.

Que 1. Consider the flow network in Figure (1a) with a source s and a sink t. The edge capacities are shown on the edges. Figure (1b) shows an s-t flow in the same network with flow values indicated on the edges.



(a) A flow network with capacities on the edges

(b) A flow network, with current flow values shown on the edges.

Figure 1: Network Flow

(a) [2 mark]. Construct the residual graph with respect to the flow shown in Figure (1b) and capacities shown in Figure (1a). Recall that any forward/backward edges with zero capacity are not kept in the residual graph.

(b) [1 mark]. Give a short argument to prove that maximum s-t flow value of this network (capacities in Figure (1a)) is 30.

(c) [1 marks]. You are allowed to increase capacity of any one edge by 1 unit. Which edge will you choose so that the maximum flow value will increase by 1 unit.

Ans 1. (a)

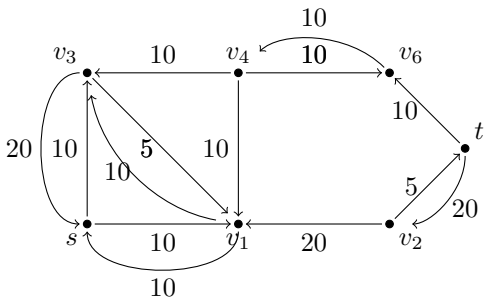


Figure 2: Residual network with residual capacities on the edges

If one has shown only forward edges, 0.5 out of 2 marks.

(b). In the given flow assignment (Figure 1b), the outgoing flow from s is indeed 30. And we see that in the residual network with respect to this flow (Figure 2), there is no path from s to t. Hence, there is no way to increase the flow further.

Alternate argument. In the given flow assignment (Figure 1b), the outgoing flow from s is indeed 30. In the capacity network (Figure 1a), we see an s-t cut, $\{s, v_1, v_3\}$ whose total outgoing capacity is 30. Hence, 30 is the maximum s-t flow.

Alternatively, one can give the cut $\{s, v_1, v_3, v_4, v_6\}$ as a cut with outgoing capacity 30.

(c) [1 marks]. (v_1, v_2)

Que 2 [3 marks]. SAT stands for the satisfiability problem. Suppose a problem X is in NP. Then say True or False for each of the below.

(a) X certainly cannot be solved in polynomial time.

(b) If X can be solved in polynomial time then SAT can be solved in polynomial time.

(c) X may or may not be NP-complete.

(d) If SAT can be solved in polynomial time then X can be solved in polynomial time.

(e) To show that X is NP-complete, it is sufficient to show that X reduces to SAT.

(f) If X is NP-complete then SAT reduces to X.

Ans 2. No explanation was required.

(a) False

We know that every problem in P is also in NP.

(b) False

For example, X can be adding two numbers. How can that help with solving SAT.

(c) True

Some problems in NP are NP-complete, while some others are not (known to be)

(d) True

Since SAT is NP-complete, every problem in NP can be converted to SAT, and hence can be solved if there is an algorithm for SAT.

(e) False

To show NP-completeness, we should show that SAT (or any other NP-complete problem) reduces to X.

(f) True

By definition of NP-complete, every problem in NP, including SAT, reduces to X.

Que 3. You have collected some statistics about populations in different districts and in different age groups. These numbers are recorded in multiples of one lakh and they are written in a matrix. See the below table, for example.

	Age group 1	**Age group 2**	**Age group 3**	**Sum**
District 1	18.74	10.38	5.88	35.00
District 2	15.29	7.00	5.71	28.00
District 3	14.97	8.62	6.41	30.00
Sum	49.00	26.00	18.00	93.00

For simplicity, assume that the row sums and the column sums are all integers. The census rounding problem is to round all the entries without changing the row sums and column sums. Each fractional number can be rounded up or down. Integers must remain as it is. A good rounding scheme for above example is as follows.

	Age group 1	**Age group 2**	**Age group 3**	**Sum**
District 1	18	11	6	35.00
District 2	16	7	5	28.00
District 3	15	8	7	30.00
Sum	49.00	26.00	18.00	93.00

The goal is to prove that this is always possible. We will do this in two steps.

(a) [4 marks]. First assume that each matrix entry $x_{i,j}$ satisfies $0 \leq x_{i,j} < 1$, and row sums and column sums are integers. We want to round each nonzero entry to either 0 or 1, without changing the row sums and columns sums. Can you check whether this is possible, using a max flow computation in some network? Recall that when the capacities are integral then the max flow algorithm outputs an integer flow (each edge has a flow with integer value). Can you argue that the desired rounding is always possible?

Ans 3(a). Construct a flow network where we have one vertex for each row, say r_i and one vertex for each column, say c_j. Additionally, we have a source vertex s and a sink vertex t. From s we have an edge to each r_i, whose capacity is the row sum of the ith row. Similarly, we have an edge from each c_j to t, whose capacity is the column sum of the jth column. From r_i to c_j, add an edge with capacity 1, if and only if the (i, j) matrix entry is **nonzero.**

-1 for not taking care of the zero entries.

Note that given matrix entries give us a valid flow in the network (the edge (r_i, c_j) has flow of $x_{i,j}$ units). Here total outgoing flow from s is equal to $\sum_i (i\text{th row sum}) = \sum_j (j\text{th column sum})$. This is actually the maximum flow, because total outgoing capacity of s is the same.

If we run the maximum flow algorithm on this network, we will get an integral flow because capacities are integral. Thus, each each (r_i, c_j) will have a flow of 0 or 1 units, let us denote it by $x'_{i,j}$. By flow conservation, we see that $x'_{i,j}$ values will satisfy row sums and column sums (originally zero entries will remain zero because we did not have edges for them). Thus, we have the desired rounding.

(b) [2 marks]. Now coming to the general case when there is no bound on the matrix entries (some of the entries may be integers) and row sums and column sums are integers. Show that it is possible to round each entry α to either $\lceil \alpha \rceil$ or $\lfloor \alpha \rfloor$, without changing the row sums and column sums.

Ans 3(b). In this case, from each entry α, we subtract $\lfloor \alpha \rfloor$. In particular integral entries will become zero and non-integral entries will fall between 0 and 1. The row sums and column sums are appropriately adjusted. Now, we run the rounding from part (a). And then whatever we had subtracted from matrix entries, we add them back. Recall from part (a) that zero remains zero and any nonzero entry becomes 0 or 1. Hence, finally in our matrix, α will be rounded to either $\lceil \alpha \rceil$ or $\lfloor \alpha \rfloor$. The row sums and column sums are preserved as in part (a).

Further extension: It was not asked in the question but you can try to solve the case when row sums and columns sums are not necessarily integral. Now, we want to round the entries so that any entry α will be rounded to either $\lceil \alpha \rceil$ or $\lfloor \alpha \rfloor$ and any sum s becomes either $\lceil s \rceil$ or $\lfloor s \rfloor$. Is this possible?

Fun fact: this kind of rounding is not always possible if we have data along 3 or more axes, and to check whether it is possible is NP-hard.

Que 4. Consider a generalization of the interval scheduling problem. You have a resource and there are multiple requests to use that resource. Each request comes with a set of intervals. For example, first request wants to use the resource from 10 am to 11 am, and then from 2 pm to 3 pm. If you accept a request, then you will have to provide the resource for all the desired intervals by that request. Naturally, you can accept any two requests only if none of their intervals have any overlap.

Example:
Request 1: 10-11, 14-16
Request 2: 8-9, 15-18

Request 3: 11-12, 17-18

Here, you can accept Request 1 and Request 3.

Decision problem: given a set of requests with corresponding sets of intervals, and a number k, is it possible to accept at least k requests?

(a) [1 marks]. Show that the above problem is in NP.

(b) [6 marks]. Prove that the above problem is NP-hard. For example, you can reduce the independent set problem to the above problem.

(a). The problem is in NP as follows. For the yes inputs, the proof can be a list of k requests. The verifier can verify in polynomial time that no two requests have any overlaps in their intervals.

(b). Reduction from independent set problem to the given problem. We will start with a given graph G and compute a set of requests R_G, each with its set of intervals, such that

- if G has an independent set of size k then it is possible to accept k requests from R_G
- and if it is possible to accept k requests from R_G then G has an independent set of size k.

Let us create one request r_v for each vertex v in G. Suppose the graph has m edges. Divide the day into m disjoint intervals. For each $1 \leq j \leq m$, if the jth edge is between vertex u and v, then we insert the jth interval in the list of request r_u and also in the list of request r_v.

From the construction it is clear that there is an edge between u and v in G if and only if some interval from r_u overlaps with (is equal to) some interval from r_v (in fact, the two intervals are equal).

Suppose G has an independent set of size k. Consider the corresponding set of k requests. Since there is no edge in these k vertices, no two requests among these k will have any intervals in common (and hence, no overlaps). Thus, these k requests can be accepted together.

Suppose there are k requests which can be accepted. Then no two of these requests have any intervals overlapping. Hence, the corresponding k vertices in G have no edges among them, which gives us an independent set.

This finishes the proof of correctness.