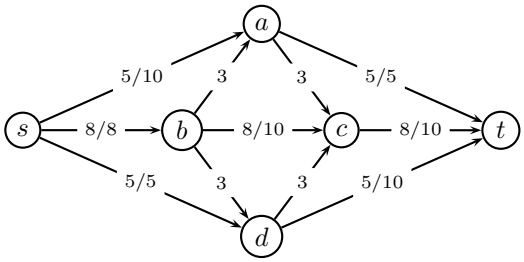
CS624: Analysis of Algorithms

Assignment 7

Due: Dec. 12, 2025

1. **Max-Flow-Min-Cut:** Given the following flow network on which an s-t flow has been computed. The capacity of each edge appears as a label on the edge, and the numbers in parentheses give the amount of flow sent on each edge. (Edges without parentheses – specifically, the four edges of capacity 3 – have no flow being sent on them.)



   (a) What is the value of this flow? Is this a maximum (s,t) flow in this graph?

   (b) Find a minimum s-t cut in the flow network and also say what its capacity is.

2. Decide whether the following statement is true or false. If it is true, give a short explanation. If it is false, give a counter example:

   Given an arbitrary flow network, with a source s, a sink t, and a positive integer capacity c_e on every edge e ;and let (A, B) be a minimum $s - t$ cut with respect to these capacities $\{c_e | e \in E\}$. Now suppose we add 1 to every capacity; then (A, B) is still a minimum $s - t$ cut with respect to these new capacities $\{1 + c_e | e \in E\}$.

3. Exercise 1.1 (page 3 in the NP notes).

4. Exercise 2.1 (page 9)

5. Exercise 3.6 (page 16)

6. In class (and in the notes) we showed that INDEPENDENT SET was NP-complete by showing that VC $\leq_P$ INDEPENDENT SET. Here is another way to do this, by showing directly that 3-SAT $\leq_P$ INDEPENDENT SET. I'm going to write most of the proof out. Your job will be to finish it up.

   Let $e = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ be a 3-SAT expression on n variables $v_1, v_2, \ldots, v_n$. (Remember that this means that each clause c_j is of the form $z_1^{(j)} \vee z_2^{(j)} \vee z_3^{(j)}$ where each $z_k^{(j)}$ is a *literal*—that is, it is either v_p or $\bar{v}_p$ for some p.)

Our task is to construct a polynomial-time mapping that maps each such e into an undirected graph G such that

$$G \text{ has an independent set of size } m \iff e \text{ is satisfiable.}$$

We can assume that each clause c_j is made out of *distinct* variables, that is, no variable occurs twice in any one clause.

First, let's take a simple example of a clause: $c = (v_1, \bar{v}_3, v_4)$. There are exactly 7 possible assignments of truth values to the variables v_1, v_3, and v_4 that give c the value T – in fact there are 8 possible assignments to the triplet $\langle v_1, v_3, v_4 \rangle$, and every one of these *except* $\langle \mathrm{F}, \mathrm{T}, \mathrm{F} \rangle$ makes c have the value T. Clearly this same reasoning applies to every clause, no matter how it is constructed.

Now we are ready to construct the graph G. Each clause c_j in e will give rise to a group of 7 vertices in G. Each of these vertices will correspond to one of the 7 possible assignments of truth values to the variables in c_j that make c_j true.

We do this for each of the clauses in e. This gives us m groups of 7 vertices each, for a total of $7m$ vertices in G.

Now for the edges: We let the edges denote "interference", or "inconsistency". That is, we will put an edge between two vertices iff those two vertices assign opposite values to some variable.

So for instance, every two vertices in any one of the groups has an edge between them. (That is, each group is a clique, although that fact is not important for us.) And there may also be other edges, going between vertices in different groups as well.

Your task then is to show the following:

(a) The transformation from e to G outlined above is a polynomial-time construction. (This is trivial really, but you have to say *something*.)

(b) e is satisfiable $\implies$ G has an independent set of size m.

(c) G has an independent set of size m $\implies$ e is satisfiable.

7. The SET-Packing problem is defined as follows: a set S together with a positive integer k. **Question:** Is there a collection C of k subsets of S that are pairwise disjoint, That is, no two subsets $c_i \in C$ and $c_j \in C$ share elements?

   (a) Show that SET-Packing is in NP.

   (b) Show that SET-Packing is NP-complete using a reduction from the INDEPENDENT-SET problem. Don't forget to show the reduction is polynomial.