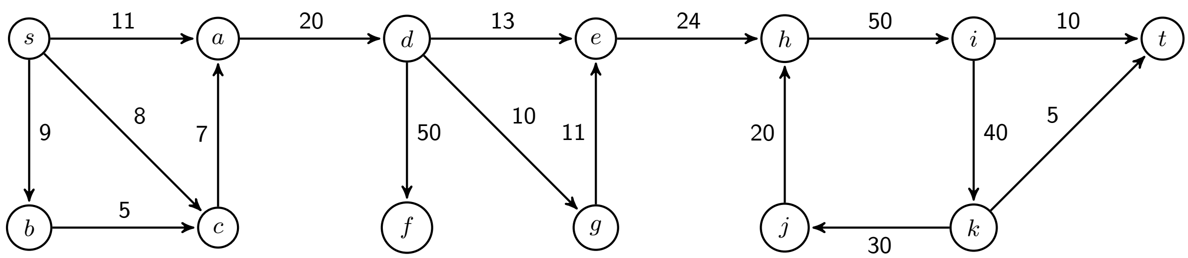
COMPSCI 311: Introduction to Algorithms	Fall 2021

Discussion 10

Your Name: ____________ Collaborators: ____________

Problem 1. Network Flow. What is the value of the maximum flow from s to t in the following graph? Use cuts to determine the answer.



Problem 2. Implications of polynomial-time reductions. Remember that Problem Y is polynomial-time reducible to Problem X if there is an algorithm for solving Problem Y that looks like this:

solveY(yInput)
  Construct xInput ▷ polynomial time
  foo = solveX(xInput) ▷ polynomial number of calls
  return yes/no based on foo ▷ polynomial number of time

This means we can solve any instance of Problem Y using a black-box solver for Problem X and at most a polynomial amount of additional work.

Last week we showed that Bipartite-Matching is polynomial-time reducible to Network-Flow (we would write this as Bipartite-Matching $\leq_P$ Network-Flow). We know that there is a polynomial-time algorithm for Network-Flow (an efficient variant the Ford-Fulkerson algorithm). What does this imply about Bipartite-Matching?

(a) There is a polynomial-time algorithm for Bipartite-Matching.

(b) There is no polynomial-time algorithm for Bipartite-Matching.

(c) Nothing

Now consider a different pair of problems: Mipartite-Batching and Fletwork-Know, where, like their counterparts above, Mipartite-Batching is polynomial-time reducible to Fletwork-Know (that is, Mipartite-Batching $\leq_P$ Fletwork-Know). Suppose now that you prove that there is *no* polynomial-time algorithm for Mipartite-Batching. What does this imply about Fletwork-Know?

(a) There is a polynomial-time algorithm for Fletwork-Know.

(b) There is no polynomial-time algorithm for Fletwork-Know.

(c) Nothing

Explain your answer.

Problem 3. Interval Scheduling. K&T Chapter 8, Exercise 1. For each of the questions below, decide whether the answer is (i) "Yes", (ii) "Unlikely, because it would show that an NP-complete problem can be solved in polynomial time, which would prove that P = NP". Explain your answer.

Let's define the decision version of the Interval Scheduling Problem from Chapter 4 as follows: Given a collection of intervals on a time-line, and a bound k, does the collection contain a subset of nonoverlapping intervals of size at least k?

(Hint: you may use the fact that Vertex Cover and Independent Set are NP-complete. Also, recall that reductions are transitive: if $Y \leq_P X$ and $X \leq_P U$, then $Y \leq_P U$.)

1. Question: Is it the case that Interval Scheduling $\leq_P$ Independent Set?
2. Question: Is it the case that Interval Scheduling $\leq_P$ Vertex Cover?
3. Question: Is it the case that Independent Set $\leq_P$ Interval Scheduling?

Problem 4. Diverse Subset. K&T Chapter 8, Exercise 2. A store trying to analyze the behavior of its customers will often maintain a two-dimensional array A, where the rows correspond to its customers and the columns correspond to the products it sells. The entry $A[i, j]$ specifies the quantity of product j that has been purchased by customer i.

Here's a tiny example of such an array A.

	detergent	beer	diapers	cat litter
Raj	0	6	0	3
Alanis	2	3	0	0
Chelsea	0	0	0	7

One thing that a store might want to do with this data is the following. Let us say that a subset S of the customers is *diverse* if no two of the of the customers in S have ever bought the same product (i.e., for each product, at most one of the customers in S has ever bought it). A diverse set of customers can be useful, for example, as a target pool for market research.

We can now define the DIVERSE-SUBSET Problem as follows: Given an $m \times n$ array A as defined above, and a number $k \leq m$, is there a subset of at least k of customers that is diverse?

Show that INDEPENDENT SET $\leq_P$ DIVERSE-SUBSET (read: INDEPENDENT SET is polynomial-time reducible to DIVERSE-SUBSET).

(Since INDEPENDENT-SET is NP-complete, this can be used to show that DIVERSE-SUBSET is also NP-complete.)