Boston University CS 630 Fall 2024
Review Problems

These are some practice problems for the upcoming midterm exam. Note that the exam is only 75 minutes (conducted during lecture time), the number of problems here far exceed the length of the actual exam. To give you an idea we have marked some problems with (*). The set of marked problems together would constitute an exam of appropriate length. The marking does not mean that we won't ask practical or theoreticl questions on one particular topic. We intend to have a good mix of the two.

How to study? Here are some recommendations on how to go about preparing for the exam.

- Know all definitions.
- Be able to run/trace algorithms on specific instances.
- As you may have noticed, most of the problems in labs and the homework were closely related to some problem from lecture. Make yourself so familiar with the lecture problems that you can comfortable make the connection between those and the problem in the assignment. (e.g. the homework problems were related to Independent Set and Set Cover respectively) This will also help you reference or apply approximation factors.
- You might find reviewing TopHat questions useful.
- Review lab and homework assignments.
- Go over the reading material (see Piazza post)
- Do a thorough job preparing the "perfect" cheat sheet. Thinking about what should exactly go on it will help you organize your knowledge.

1 Network Flow

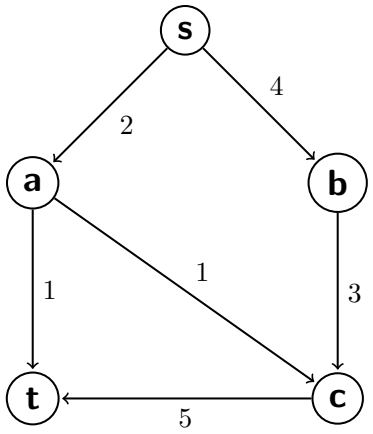
1. (*) Consider this weighted directed graph G with source s, sink t and edge capacities written on the edges.

   (a) List *all* the min-cuts in G.

   (b) Suppose that the capacity on edge (c, t) is changed to 2 (from 5). First, list the min-cuts in this modified graph. Second, answer whether the maximum flow in this graph is unique. Explain why or why not.

2. (Jeff Erickson Algorithms, page.363, exercise 5)

   (a) (*) Describe a flow network that contains a *unique maximum (s, t)-flow* but does *not* contain a *unique minimum (s, t)-cut*.



(b) (*) Describe a flow network that contains a *unique minimum (s, t)-cut* but does *not* contain a *unique maximum (s, t)-flow.*

(c) Describe an efficient algorithm to determine whether a given flow network contains a *unique minimum (s, t)-cut.*

3. What is the tightest bound you can give on the running time of the Ford-Fulkerson algorithm when the input graph satisfies the following: all vertices have degree at most D_{max}, and all edges have capacity at most C_{max} (where C_{max} is an integer)? Your bound should depend on n, C_{max} and D_{max} but nothing else.

4. Kleinberg-Tardos, Chapter 7, exercises 1–9 (page 415+).

5. Kleinberg-Tardos, Chapter 7, exercise 15 (page 421).

6. Kleinberg-Tardos, Chapter 7, exercise 16 (page 422).

7. (*) Kleinberg-Tardos, Chapter 7, exercise 19 (page 425).

8. (*Note that if you were to get a problem like this, you don't have to do it from scratch, you could reference relevant ideas from the homework!*) Suppose you have already computed a maximum-value s-t-flow f for a graph $G = (V, E, c)$. Describe how, given $G, s, t, f, (u, v)$, you would update the flow in $O(m + n)$ time if the following changes were made to G:

   (a) First, how can you verify in time $O(n + m)$ that the flow is indeed maximum?

   (b) The edge (u, v) with capacity 1 is added to the graph.

   (c) The edge (u, v) with capacity 1 is deleted from the graph.

   (d) The capacity of an existing edge (u, v) is *increased* by 1.

(e) The capacity of an existing edge (u, v) is *decreased* by 1.

9. During the pandemic, there is a global push to vaccinate as many people as quickly as possible. There are K different brands of vaccine available that need to be distributed to L countries. For each vaccine v_i there are q_i doses available. Further, we know the population p_j of each country c_j. However, not every vaccine is approved for use in every country. Currently, for each vaccine v_i there is a list ℓ_i of countries where it is approved.

   Find an algorithm to decide whether there is a way to immunize every person in every country with a vaccine approved by their country's health department.

   Your algorithm should use a reduction to network flow. Specify your algorithm by answering the questions below. Your answers should be clear enough that any other student could turn them in to working code. (Pictures are helpful.)

   (a) Design a directed graph $G(V, E)$ that is an instance of the max-flow problem. That is, G is directed, it has designated source s and sink t nodes and a capacity $c(e)$ on each directed edge. Describe each part of G:
   - Specify the vertices in G. *Hint: there should be nodes for both vaccines and countries.*
   - Describe the edges in G. Make clear which direction they are.
   - Describe the capacity on each edge.
   - Make sure there are vertices named s (source) and t (sink)

   (b) How many vertices does your graph have in terms of K and L (asymptotically)?

   (c) What is the maximum number of edges your graph can have, in terms of K and L (asymptotically)?

   (d) Suppose we found the maximum flow f in G. Describe how to find for each i, j the number of units of the vaccine v_i to be sent to country c_j given the value $f(e)$ along each edge.

   (e) Suppose that it is indeed possible to get each country its required amount of vaccine. In that case, what is the value of the flow that is returned by the max-flow subroutine?

2 NP, NP-C and polynomial time reductions

1. (* there would be one such question the exam.) Prove that the following problems are in NP. You will do this by (0.) Formulate the problem as a decision problem, i.e. it has a yes/no answer. (1.) Define the polynomial-length certificate and the polynomial-time verifier function that takes the problem and certificate as input. (2.) Show how the certificate correctly identifies the problem instances with "yes" answers.

   (a) Perfect Matching

   (b) Factoring

(c) Longest Path

(d) Hamiltonian-Path, Hamiltonian-Cycle, Traveling Sales Person (TSP)

(e) Independent Set, Vertex Cover, Set Cover

(f) Graph k-coloring (Given a graph G, can we assign at most k colors to its vertices, so that neighbors have different colors?)

(g) 3-SAT

2. (* we would ask 4-5 such questions on the exam.) For each of the following statements, indicate whether it is true, false, or "unknown", where "unknown" indicates that scientists have not conclusively determined whether the statement is true or false. For example, the correct answer for the statement "$P = NP$" is "unknown". But the correct answer for "the best algorithm for the maximum flow problem in n-vertex graphs takes time at least 2^n" is "false".

   Give a short justification for your answer (i.e. explain why the statement is true, or false, or not known to be either true or false).

   (a) If X is in NP, then there does not exist a polynomial-time algorithm for X.

   (b) If X is NP-complete, then there is no polynomial-time algorithm for X.

   (c) If X is NP-complete, Y is in NP, and X reduces to Y (thus $X \leq_p Y$), then Y is NP-complete.

   (d) Shortest Paths is NP-complete.

   (e) Shortest Paths is in NP.

   (f) Shortest Paths is in P.

   (g) Longest Paths is NP-complete.

   (h) Longest Paths is in NP.

   (i) Longest Paths is in P.

   (j) Every problem in P is in NP.

   (k) Every NP-complete problem is in NP.

   (l) Some NP-complete problems have polynomial time algorithms, but some others do not.

(m) Suppose that X is polynomial and $X \leq_P Y$, then Y is a polynomial problem.

(n) Suppose that X is exponential, and $X \leq_P Y$, then Y is at least exponential.

(o) Suppose that X is in NP, and $X \leq_P Y$, then Y is NP-C.

(p) Suppose that Y is polynomial, and $X \leq_P Y$, then X is polynomial.

3. In a directed graph G the Directed Hamiltonian-Path (DHP) problem decides whether there is a simple path in G that contains every vertex. Show that DHP is NP-C. (Don't forget to show that DHP is in NP and a reduction from some other known NP-C problem *to* DHP.)

4. The Clique problem in an undirected graph G finds the largest clique subgraph in G. (A clique is a complete graph, i.e. a graph such that every vertex is connected to every other vrtex.) Show that this problem is NP-C. (*Hint: you might want to look at the complement of this graph*)

3 Approximation Algorithms

1. (*) Answer the following questions and give a short explanation.

   (a) Suppose that some algorithm is a 3-approximation algorithm to a minimization problem. This algorithm outputs on some input a solution of value 120. What is the range of value for the optimal solution on this input?

   (b) Suppose that there is a c-approximation algorithm $myAlgo$ for problem X. Then for any input to $myAlgo$ the output is c-times worse than the optimal solution for this particular problem instance.

   (c) There are no approximation algorithms for polynomial problems.

   (d) Suppose that there is a polynomial time reduction from problem X to a problem Y. Further, we have a c-approximation algorithm for Y. Then this algorithm applied to (the reduced form of) problem X yield a c-approximation for X.

   (e) If the c-approximation for Y in the above problem is tight, i.e. there is no algorithm for Y with a better approximation ratio, then this ratio is also tight for X.

2. Consider the following algorithm for Bin Packing; items are considered in the fixed input order. The next item will be assigned to the most recent bin, if it doesn't fit then a new

bin is started. Show that this is a 2-approximation algorithm. *Hint: look at the total sum of elements and the number of bins used.*

3. Describe a 2-approximation algorithm for the following Number Multi-Partitioning problem: We are given as input n numbers $x_1, x_2, \ldots, x_n$ and an integer m. Divide the numbers into m sets, such that the total value in the set with the largest sum is minimized. (*Hint: this should remind you very much of one of the problems that we studied.*)

4. (*) Consider the Restricted Length Load Balancing problem; we want to run n jobs, each of them has a length t_j. We have unlimited machines available, however all the jobs have to be finished within 24 hours. Find an assignment of jobs to machines so that all jobs are finished on time and as few machines are used as possible. Find a 2-approximation algorithm.