Final for CS 170

PRINT your name: Myers (last) , Gene (first)

Three pages of notes are permitted. Two from your prior midterms and a new one prepared specifically for the final. No electronic devices, e.g. cell phones and calculators, are permitted. Do all your work on the pages of this examination. If you need more space, you may use the reverse side of the page, but try to use the reverse of the same page where the problem is stated.

You have 3 hours. The questions are of varying difficulty, so avoid spending too long on any one question.

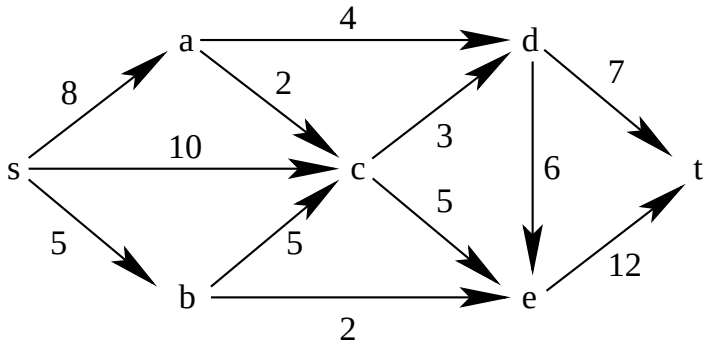
In all algorithm design problems, you may use high-level pseudocode.

DO NOT TURN THE PAGE UNTIL YOU ARE TOLD TO DO SO.

Problem	Score/Points
1	/10
2	/10
3	/10
4	/10
5	/10
6	/10
Total	/60

1. Network Flow

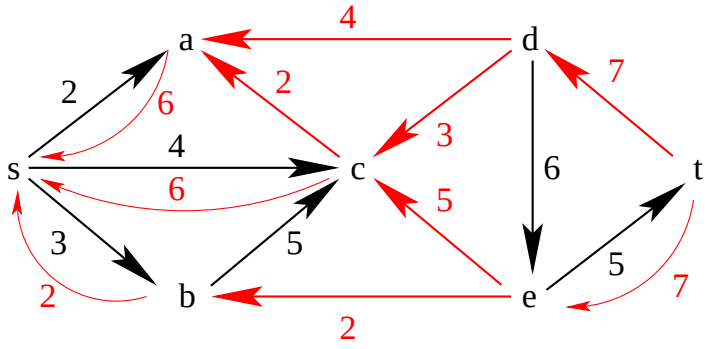
Find the maximal flow in the following network. Give each augmenting path and the net flow gain obtained (we give the first one). Use the back sheet of the previous page for your work.



Augmenting Path	Net Flow Gain
s - a - d - t	4
s - a - c - d - t	2
s - c - d - t	1
s - c - d - t	5
s - b - e - t	2
Total	14

Draw the residual graph at the conclusion of the algorithm, and draw or list a minimum cut of the graph.

Augmenting Path	Net Flow Gain
a - d	4
c - d	3
c - e	5
b - e	2
Total	14



2. Linear Programming

The Canine Products company has two dogfood products, Frisky Pup and Husky Hounds, that are made from a blend of two raw materials, cereal and meat. 1 pound of cereal and 1.5 pounds of meat are needed to make a package of Frisky Pup and it sells for \$7 a package. 2 pounds of cereal and 1 pound of meat are needed to make a package of Husky Hound and it sells for \$6 a package. Raw cereal costs \$1 per pound and raw meat costs \$2 per pound. It also costs \$1.40 to package the Frisky Pup and \$.60 to package the Husky Hound. A total of 240,000 pounds of cereal and 180,000 pounds of meat are available per month. The only production bottleneck is that the factory can only package 110,000 bags of Frisky Pup per month. Management would like to maximize profit.

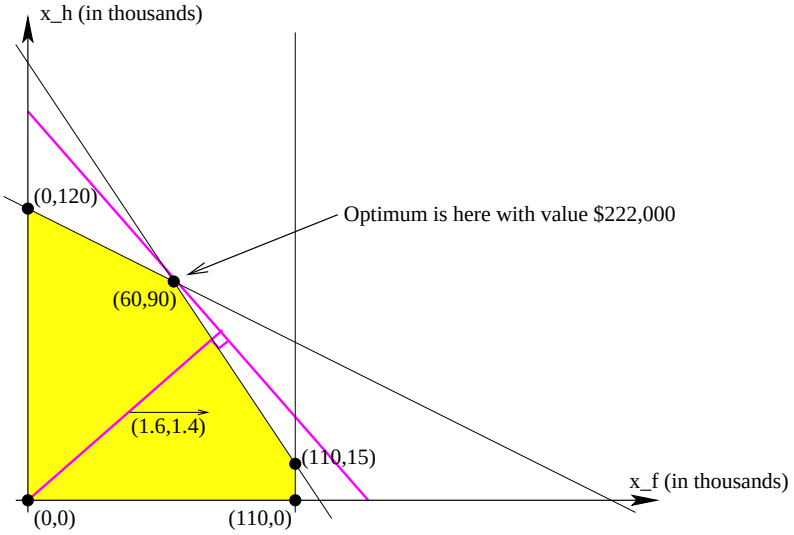
(a) Formulate the problem as a linear program, be sure to explain your formulation by stating what each variable represents, the reason for each contraint, and how the objective was formulated.

Let x_S be the number of packages of product S produced per month where $f \equiv$ Frisky Pup and $h \equiv$ Husky Hound. Figuring the sales price less the cost of packaging and raw inputs, the profit on each package of Frisky Pup is \$1.60 and \$1.40 for Husky Hound. Thus we have:

$\max 1.6x_f + 1.4x_h$ (total profit) subject to

$$\begin{array}{rcrl} 1x_f + 2x_h & \leq & 240{,}000 & \text{(constrained imposed by available amount of cereal)} \\ 1.5x_f + 1x_h & \leq & 180{,}000 & \text{(constrained imposed by available amount of raw meat)} \\ x_f & \leq & 110{,}000 & \text{(constraint imposed on package capacity for Frisky Pup)} \\ x_f, x_h & \geq & 0 & \text{(producing a negative number of packages would be non-sensical)} \end{array}$$

(b) Graph the feasible region, give the coordinates of every extermal point, and circle the extremal point maximizing profit. What is the maximum profit possible?



(c) Give the dual formulation of your LP problem. What is the intuitive meaning of each of the dual variables?

We view the dual LP as a valuation problem and ask what the value per pound of cereal, y_c, the value per pound of meat, y_m, and the value per package of Frisky Pup packaging, y_p.

$\min 240,000y_c + 180,000y_m + 110,000y_p$ (total value) subject to

$$\begin{array}{rcll} 1y_c + 1.5y_m + 1y_p & \geq & 1.6 & \text{(value of a package of Frisky Pup)} \\ 2y_c + 1y_m & \geq & 1.4 & \text{(value of a package of Husky Hound)} \\ y_1, y_2, y_3 & \geq & 0 & \text{(negative valuations make no sense)} \end{array}$$

3. Bottleneck Shortest Paths

Consider a weighted directed graph $G = (V, E)$ where the weight $w(u \to v)$ on any edge $u \to v$ is a positive integer. Let the *bottleneck length* of a path be the weight of the edge in the path with the greatest weight. Let the *bottleneck distance* $b(v, w)$ from v to w be the minimum of the bottleneck lengths of all paths from v to w. For a given source vertex s, design a single-source shortest bottleneck path algorithm, i.e. one that computes $b(s, v)$ for all v. Be sure to justify the correctness of your algorithm, give a pseudo-code formulation of it, and state and justify its worst-case time complexity.

Design/Correctness: The primary observation is that for any given distance d, all vertices of distance d or less from s are exactly those reachable from s along the subgraph of all edges of weight not greater than d. Thus we seek to compute the distance to vertices in increasing order of distance and clearly it suffice to examine the weights of the edges as every distance will correspond to one of these E values.

We maintain a heap such that at the start of each iteration the following inductive invariant is true: If the smallest edge in the heap has weight d, then every vertex of distance less than d has been found, and all the edges of weight d-or-more whose source is one of these vertices is in the heap. In an iteration of the algorithm, we extract the next minimum edge, determine all the vertices that can be reached through that edge along edges of equal or lesser score, and update the induction.

```
procedure Search(v: vertex, d: weight)
  {  D[v] ← d
     for v → u ∈ E do
          if w(v → u) ≤ d and D[u] = ∞ then
               Search(u,d)
          else
               Add v → u to Heap
  }

D[v] ← ∞  ∀v
D[s] ← 0
Heap ← {s → v}
while Heap ≠ ∅ do
  {  e(≡ v → u) ← extract min Heap
     if D[u] = ∞ then
          Search(u, w(e))
  }
```

Complexity: An edge is added to the heap only if its source has just been assigned its distance (if then). Therefore each edge is added to the heap at most once. Moreover, Search is only called when a vertex is about to be assigned its and so is not called more than V times. The algorithm is thus $O(E \log V)$ in the worst case.

4. Finding a Path of a Specific Weight

Consider a weighted directed graph $G = (V, E)$ where the weight $w(u \rightarrow v)$ on any edge $u \rightarrow v$ is a positive integer. For a given source s, sink t, and integer $k > 0$, design a dynamic programming algorithm that determines if there is a path (it need not be simple) from s to t of weight exactly k.

(a) Formally define the set of sub-problems you will solve.

For $v \in V$ and $i \in [0, k]$, let:

$Path[v, i] = \exists$ path from s to v of weight i.

(b) Give your recurrence for the solution of a given sub-problem in terms of other sub-problems.

$$Path[v, i] = \begin{cases} OR_{u \rightarrow v} \;\; Path[u, i - w(u \rightarrow v)] & \text{if } i > 0 \\ v = s & \text{if } i = 0 \end{cases}$$

(c) Give a non-recursive pseudo-code specification of the algorithm and state its complexity in terms of V, E, and k.

```
Path[s, 0] ←true
for v ∈ V − {s} do
    Path[v, 0] ←false
for i ← 1 . . . k do
    for v ∈ V do
     {  Path[v, i] ← false
        for u → v ∈ E do
            if Path[u, i − w(u → v)] then
                Path[v, i] ← true
     }
if Path[t, k] then
    print "Yes"
else
    print "No"
```

By inspection the algorithm takes $O((V + E)k)$ time.

5. NP-Completeness

Consider an undirected graph $G = (V, E)$. A *k-coloring* of the graph is an assignment of k distinct colors to the vertices of the graph in such a way that no two vertices assigned the same color are adjacent (i.e. have an edge between them). The GRAPH-COLORABILITY problem is to find a coloring involving the fewest colors.

(a) State precisely the decision version of the GRAPH-COLORABILITY problem.

Given an undirected graph G and integer $k > 0$, can the graph be colored with just k colors?

(b) Show that it is in NP.

One can guess a coloring, by producing a list of the vertices and its associated color assignment. This guess, or verification certificate, is certainly polynomial in the size of G. Verification simply entails checking that only k colors are assigned, and examing every edge of the graph and checking to see if the colors assigned to the vertices at each end are distinct. This is easily accomplished in $O(V + E)$ time.

(c) Next we show that GRAPH-COLORABILITY is NP-hard by giving a polynomial reduction from 3-SAT (i.e. is a given 3-CNF formula satisfiable) to GRAPH-COLORABILITY. Suppose you are given a formula with m clauses C_i and n variables x_i. Construct a graph with $3n + m$ vertices $v_1, v_2, \ldots v_n$, $x_1, x_2, \ldots x_n$, $\bar{x}_1, \bar{x}_2, \ldots \bar{x}_n$, and $c_1, c_2, \ldots, c_m$. Add exactly the following edges:

(a) (v_i, v_j) for all $i \neq j$.

(b) (v_i, x_j) and $(v_i, \bar{x}_j)$ for all $i \neq j$.

(c) $(x_i, \bar{x}_i)$ for all i.

(d) (x_i, c_j) if x_i is *not* a term in clause C_j, and $(\bar{x}_i, c_j)$ if $\bar{x}_i$ is *not* a term in clause C_j.

Prove that for $n \geq 4$, the constructed graph is $n + 1$ colorable if and only if the 3-CNF from which it is constructed is satisfiable. (*Hint:* The v_i vertices form a complete graph or clique of n vertices and so must each be assigned a distinct color. Think about what to do with your one extra color.)

First suppose that the 3-SAT formula is satisfiable. We show that we can color the constructed graph with $n + 1$ colors. Assign a distinct color to each of the v_i. For each variable x_i, if it is true, then assign vertex x_i the same color as v_i and assign $\bar{x}_i$ the $n + 1^{st}$ color, which we call the *special color*. Switch the assignment of colors if x_i is false. Since the formula is satisfiable, for each clause C_j there is at least one literal that is true and the corresponding vertex was assigned a color other than the special color. Color vertex c_j with that color. c_j is not adjacent to that vertex, and no other literal vertex has the color. Thus we have produced an $n + 1$ coloring.

Now suppose that there is an $(n + 1)$ coloring of the constructed graph. Each of the v_i must be assigned a unique color as their subgraph is a clique. The vertices x_i and $\bar{x}_i$ cannot have the same color, nor the color of any v_j except for v_i. Therefore one has the color of v_i and the other has the $(n + 1)^{st}$ color, which we call the *special color*. If x_i has the color of v_i, set x_i to true, otherwise set it to false. It remains only to show that each clause is satisfied by this assignment. If C_j were false, then c_j is connected to three x vertices, corresponding to the complement of the literals in the clause, that do not have the special color, and to $2(n - 3)$ pairs of x vertices corresponding to the $n - 3$ variables not in the clause, of which $n - 3$ are non-special and at least one is the special color as $n \geq 4$. Thus C_j is adjacent to $n + 1$ colors and would have to use an additional color. Therefore C_j must be true.

6. True/False

For each statement below, say whether it is true or false, and a one sentence and/or one picture explanation.

(a) $3x = 5 (\text{mod } 22)$ has a unique solution among the integers in $[0, 21]$.

True. 3 and 22 are relatively prime.

(b) If $a^{p-1} \equiv 1 (\text{mod } p)$ for all $a \neq 0 (\text{mod } p)$, then p is prime.

False. Charmichael numbers, which are composite, also satisfy the hypothesis.

(c) If the feasible region for a linear program is unbounded, then there is no solution.

False. min x subject to $x \geq 0$ has an unbounded feasible region with a unique solution.

(d) Forward edges are impossible in a breadth first search.

True. All vertices adjacent to a vertex are examined when it is popped from the stack.

(e) Integer linear programming is NP-hard.

True. We saw that we could solve vertex cover, an NP-complete problem, with ILP.

(f) The weight of a minimum spanning tree in a positively weighted undirected graph is always less than the weight of a minimum spanning cycle (Hamiltonian Tour) of the graph.

True. Deleting any edge from a Hamiltonian Tour gives a spanning tree.

(g) If A is in NP, and B is NP-complete, and $A \leq_p B$ then A is NP-complete.

False. One has to show that $B \leq_p A$, i.e. that A is harder than B.

(h) The Lempel-Ziv compression algorithm as presented in class always returns a file shorter than the one submitted to it.

False. The coding of '0' is '00' assuming a dictionary of minimal size 1!

(i) All NP-complete problems are equally hard to approximate in polynomial time.

False. Vertex cover has $\rho = 2$ while maximum clique has $\rho > n^{.99}$.

(j) If all capacities in a network flow are rational numbers, then the maximum flow will be a rational number. (A rational number is the ratio p/q of two integers.)

True. Max flow equal min cut equals the sum of edge capacities equals a rational number.