NAME: _____

CSE 421
Introduction to Algorithms
Final Exam Winter 2005

P. Beame 14 March 2005

DIRECTIONS:

- Answer the problems on the exam paper.
- Open book. Open notes.
- If you need extra space use the back of a page
- You have 1 hour and 50 minutes to complete the exam.
- Please do not turn the page until you are instructed to do so.
- Good Luck!

1	/33
2	/20
3	/18
4	/20
5	/20
6	/20
7	/20
8	/10
Total	/161

1. (33 points, 3 each) For each of the following problems circle **True** or **False**. You do not need to justify your answer.

   (a) If f and g are two different flows on the same flow graph (G, s, t) and if $\nu(f) \geq \nu(g)$ then every edge e in G has $f(e) \geq g(e)$. **True** **False**

   (b) If f is a maximum flow on a flow graph $(G = (V, E), s, t)$ and B is the set of vertices in V that can reach t in the residual graph G_f then $(V - B, B)$ is a minimum capacity s-t cut in G.
   True **False**

   (c) If f is a maximum flow on a flow graph (G, s, t) and (S, T) is a minimum capacity s-t cut in G then *every* edge e having endpoints on different sides of (S, T) has $f(e)$ equal to the capacity of e. **True** **False**

   (d) If problem B is NP-hard and $A \leq_P B$ then A is NP-hard. **True** **False**

   (e) If problem A is NP-hard and $A \leq_P B$ then B is NP-hard. **True** **False**

   (f) If $P \neq NP$ then every problem in NP requires exponential time. **True** **False**

   (g) If problem A is in P then $A \leq_P B$ for every problem B in NP. **True** **False**

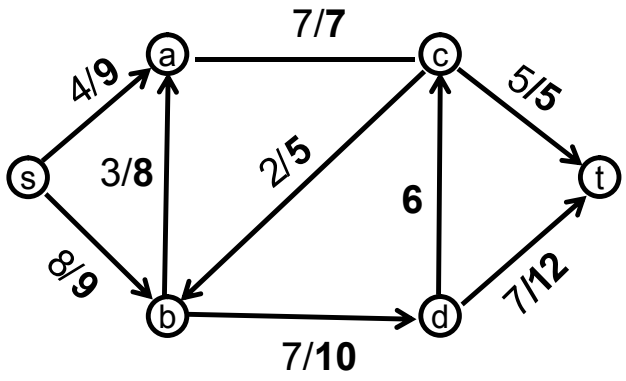
   (h) If G is a weighted graph with n vertices and m edges that does *not* contain negative-weight cycle, then the iteration of the Bellman-Ford algorithm will reach a fixed point in at most $n - 1$ rounds. **True** **False**

   (i) If G is a weighted graph with n vertices and m edges that *does* contain negative-weight cycle, then for *every* vertex v in G the shortest path from v to t in G containing n edges is strictly shorter than the shortest path from v to t in G containing $n - 1$ edges.
   True **False**

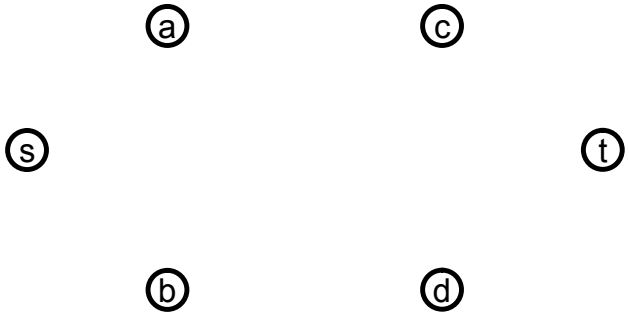
   (j) A divide and conquer algorithm for $n \times n$ matrix multiplication that solves the problem using 24 matrix multiplication subproblems of size $n/3$ and $O(n^2)$ additional work for recombining would be more efficient than the usual $\Theta(n^3)$ matrix multiplication algorithm.
   True **False**

   (k) A divide and conquer algorithm for the selection problem that reduced the problem on lists of size n to one selection problem of size $3n/20$ and another selection problem of size $9n/10$ plus a linear amount of extra work would yield a linear time algorithm.
   True **False**

2. (20 points) Consider the following flow network with a flow f shown. An edge labelled "b" means that it has capacity b and flow 0. An edge labelled "a/b" means that the flow on that edge is a and the capacity is b.

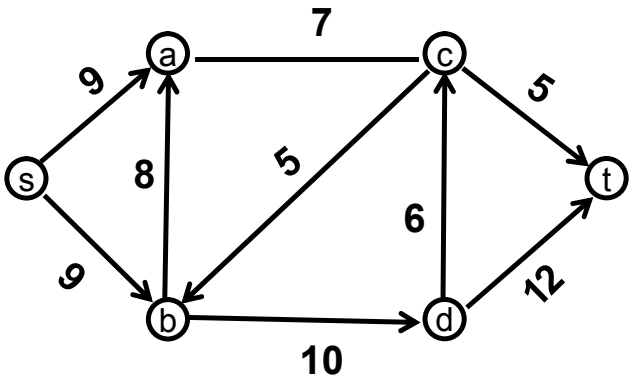


(a) (10 points) Draw the residual graph G_f.



(b) (3 points) What augmenting path in this graph would result in the greatest increase in flow value? (List the names of the vertices on this path in order.)

(c) (7 points) On the diagram below, indicate the new flow values resulting from augmenting along the path you found in part (b).



3. (18 points) For each of the following recurrences give the time and space that would be required by a simple dynamic programming algorithm to compute the answer where the values of the w function are given as input.

   (a) Compute $OPT(n)$ where $OPT(1) = 1$ and $OPT(i) = \min_{1 \leq j < i}\{OPT(j)/j + w(j)\}$.

   Time O( ) Space O( )

   (b) Compute $OPT(m, n)$ where $OPT(i, 0) = OPT(0, j) = 0$ for all i and j and $OPT(i, j) = \min\{OPT(i-1, j) + 2,\ OPT(i, j-1) + 1,\ OPT(i-1, j-1)/2 + w(j)\}$ for $i, j > 0$.

   Time O( ) Space O( )

   (c) Compute $OPT(1, n)$ where $OPT(i, i) = 0$ for all i, and $OPT(i, j) = \min\{OPT(i, k) + OPT(k+1, j) + w(k)\ :\ i \leq k < j\}$ for all $i < j$.

   Time O( ) Space O( )

4. (20 points) You are given a sequence of n bits $x_1, \ldots, x_n \in \{0, 1\}$. Your output is to be either

   - any i such that $x_i = 1$ or
   - the value 0 if the input is all 0's.

   The only operation you are allowed to use to access the inputs is a function **Group-Test** where

$$\textbf{Group-Test}(i, j) = \begin{cases} 1 & \text{if some bit in } x_i, \ldots, x_j \text{ has value 1} \\ 0 & \text{if all bits in } x_i, \ldots, x_j \text{ have value 0} \end{cases}$$

   (Historical Note: In World War I, the army was testing recruits for syphilis which was rare but required a time-consuming though accurate blood test. They realized that they could pool the blood from several recruits at once and save time by eliminating large groups of recruits who didn't have syphilis.)

   (a) (15 points) Design a divide and conquer algorithm to solve the problem that uses only $O(\log n)$ calls to **Group-Test** in the worst case. Your algorithm should *never* access the x_i directly.

   (b) (5 points) Briefly justify your bound on the number of calls.

5. (20 points) In classifying butterflies of several species, a biologist makes time-consuming comparisons between butterfly features to determine whether or not two butterflies are the same species. A biologist wants to determine as efficiently as possible whether or not there is a *dominant species* among n butterfly specimens, $B = (b_1, \ldots, b_n)$. (A dominant species is one that occurs strictly more than $n/2$ times in the list of specimens.)

   (a) Describe a divide and conquer algorithm $Dominant(B, n)$ to tell the biologist how to do this in $O(n \log n)$ applications of operation $Same(b, b')$ which returns **true** if b and b' are the same species and **false** otherwise.

   (b) Explain why your solution is correct.

   (c) Write the recurrence that justifies your claim for the $O(n \log n)$ running time.

6. (20 points) You are given a *directed acyclic graph* $G = (V, E)$ and a node node $t \in V$. Design a linear time algorithm to compute for each vertex $v \in V$ the *number* of different paths from v to t in G. Analyze its running time in terms of $n = |V|$ and $m = |E|$.

7. (20 points) The *Number Partition* problem asks, given a collection of integers (which can be positive or negative) $y_1, \ldots, y_n$ whether or not it is possible to partition these numbers into two groups so that the sum in each group is the same. Prove that *Number Partition* is NP-complete by solving the following problems.

   (a) Show that *Number Partition* is in NP.

   (b) Show that *Subset Sum* $\leq_P$ *Number Partition*
   Recall that in the *Subset Sum* problem we are given a collection of integers (which can be positive and negative), we want to see if it is possible find a subset that sum up to 1.
   Hint: Given an input to *Subset Sum* include two large numbers whose size differs by $S - 2$ where S is the sum of all input numbers.

8. (10 points) The *two processor* interval scheduling problem takes as input a sequence of request intervals $(s_1, f_1), \ldots, (s_n, f_n)$ just like the unweighted interval scheduling problem except that it produces *two* disjoint subsets $A_1, A_2 \subset [n]$ such that all requests in A_1 are compatible with each other and all requests in A_2 are compatible with each other and $|A_1 \cup A_2|$ is as large as possible. (A_1 might contain requests that are incompatible with requests in A_2.) Does the following greedy algorithm produce optimal results? If yes argue why it does; if no produce a counterexample.

```
Sort requests by increasing finish time
A_1 = ∅
A_2 = ∅
while there is any request (s_i, f_i) compatible with either A_1 or A_2 do
    Add the first unused request, if any, compatible with A_1 to A_1.
    Add the first unused request, if any, compatible with A_2 to A_2.
end while
```