CSEP 521 – Applied Algorithm
Spring 2003
Final Exam - solutions

__**Question 1 (25 points)**__

1.a Prove that the number of vertices with odd degree in a tree T=(V,E) is even.

Answer: This is true for any graph, in particular for trees.
Let x denote the number of vertices with odd degree.
The total degree of all the vertices in T is 2|E| which is even.
Let total(odd) denote the total degree of vertices with odd degree in T.
Let total(even) denote the total degree of vertices with even degree in T.
Clearly, total(even) must be even (as sum of even numbers).
This implies that the total(odd) degree must also be even (since 2|E| which is even equals total(even) + total(odd).
We know that total(odd) is a sum of x odd numbers. Such a sum is even if and only if x is even.

Another answer: A more complex proof is by induction on n, the number of vertices in the tree. **Base:** n=1, one vertex with degree 0, the claim holds.
Step: Assume that the claim is true for any tree of size <n, let T be a tree of size n.
As a tree, T has n vertices and n-1 edges. The total degree of the vertices of T is 2|E|=2n-2; thus, there must be a vertex of degree 1 in T. Let v be such a vertex.
Let u be the single neighbor of v in T.
Consider the graph T' obtained from T by removing v and the edge u-v.
T' is a tree – since it is connected, has n-1 vertices and n-2 edges.
Consider two cases:

1. in T' the degree of u is odd – thus, in T its degree is even.
2. in T' the degree of u is even – thus, in T its degree is odd.

Note that in T the vertex v has an odd degree (=1).
In case 1, the number of odd degree vertices in T equals their number in T'.
In case 2, the number of odd degree vertices in T equals their number in T'+2.
In both cases their parity is the same, which by the induction hypothesis is even.

1.b We used this property for Approximation algorithm for the traveling salesman problem

1.c Why did we need this property in the above? We need to match these vertices in pairs. By doing this, their degree will become even and we'll be able to find an Euler cycle that passes through all the vertices of G (A graph contains an Euler cycle if and only if all verices have an even degree.

<u>Question 2</u> (42 pts – *7 points each, 3 for True/False classification, 4 for explanation)*
For each of the 6 statements below. If it is true, justify it in the provided space. If it is false, give a counter example.

a. If all the capacities in a network are identical then any execution of Ford and Fulkerson algorithm will work in polynomial time.

True: Let c be the value of the identical capacities. All edges in the residual graphs to be built by F&F must also have capacity c. Thus, in each iteration we improve the value of the flow by c, and the number of iterations is the number of out-edges in the min-cut.

b. If a problem Q is NP-hard, and someone finds a polynomial-time solution to Q, then we will have a polynomial time algorithm for the 3-SAT problem.

True: By definition of NP-hardness, Q $\leq_p$ R for any NP-hard problem R, in particular 3-SAT. In other words, once we have a poly-time algorithm for Q we can use it as a black box to solve any other NP-hard problem.

c. If the Huffman code of some character is ‘0’ or ‘1’ then this character’s frequency in the code is at least 50%.

False.
Example 1: ‘a’ has freq. 1/100, ‘b’ has freq. 99/100 , an optimal coding is ‘a’ gets‘0’ and ‘b’ gets ‘1’.
Example 2; ‘a’ ‘b’ and ‘c’ each has freq. 1/3. an optimal coding: ‘a’-‘0’, ‘b’-‘10’, c-’11’.

d. In a graph G=(V,E), if the shortest distance between vertices A and B is a single edge connecting them, this edge is guaranteed to be in some minimum spanning tree of G.

False. Consider a cycle of 4 vertices a-b-c-d-a and w(a,b)=w(b,c)=w(c,d)=1, w(d,a)=2. The shortest path connecting a and d is the edge (a,d), the single MST is a-b-c-d.

e. If we run BFS on a tree, then the set of vertices with an odd label form a vertex cover whose size is at most 2 times the minimum vertex cover.

False. Consider a star, and start the BFS in the center vertex.

f. If P ≠ NP and Q is an offline NP-hard problem then no poly-time online algorithm for Q can be 1-competitive.

True: If an algorithm is 1-competitive then it is also an optimal poly-time offline algorithm.

<u>Question 3 (18 points)</u>

Describe the following problem as an *integer programming* problem:
There are m identical processors, and n jobs; job j requires p_j processing units.
Any job can be assigned to any of the processors.
Each processor can process at most three jobs (one after the other).
The goal is to assign the jobs to the processors in a way that minimizes the makespan of the schedule (the completion time of the last job).

Hint: use mn+1 variables, one of them will be denoted 't' and it should appear both in the objective function and in the constraints. Complete the IP below:

Min t (that's all the objective function)

s.t: $\Sigma_j x_{ij} \leq 3$ for all i
$\Sigma_j x_{ij}p_j - t \leq 0$ for all i
$\Sigma_i x_{ij} = 1$ for all j
$x_{ij} \in \{0,1\}$

x_{ij} indicates if job j is assigned to processor i. t is the makespan we want to minimize.

<u>Question 4 (25 points)</u>

The input to the 3-coloring problem is a graph *G*, and the problem is to decide whether the vertices of *G* can be colored with three colors such that no pair of adjacent vertices is colored the same color. The input to the 4-coloring problem is a graph *G*, and the problem is to decide whether the vertices of *G* can be colored with four colors such that no pair of adjacent vertices is colored the same color.
It is known that 3-coloring is NP-hard.

Prove that 4-coloring is NP-complete.

1. 4-coloring is in P: Given a coloring it can be verified in poly-time that any two adjacent vertices have different colors and that at most 4 colors participates.
2. 4-coloring is NP-hard: We show a reduction from 3-coloring: Assume we have a black box for 4-coloring, given an input ,G, for 3-coloring we build a graph G' obtained from G by adding one vertex which is connected to all the vertices of G.
   Formally, $G'=(V',E')$. $V'= V \cup\{v\}$ $E'=E \cup \{(v,u)| u \text{ in } V\}$
   Claim: G' is 4 colorable if and only if G is 3-colorable.
   Proof:
   1. Assume that G is colored with 3 colors, then we can color v with a fourth color to get a legal 4-coloring of G'.
   2. Assume that G' is colored with 4 colors, then no other vertex is colored like v (as v is connected to all the vertices), inducing a legal 3-coloring of G.