Final Exam
Due March 14, 2005; 5:00pm

Prof. Amit Chakrabarti
Computer Science
Dartmouth College

* Please attach this sheet to your submission. *

Your name: _____

Important Points to Note

- This exam is due by **5:00pm sharp** on **Monday, March 14**, in Prof. Chakrabarti's mailbox.
- There are **6 sections** for a total of **100 points**.
- Please **typeset** your solutions using either LaTeX or TeX and submit a printout, plus this sheet.
- Use **at most ten pages** (= five sheets, letter or A4) for the entire exam. This is a strict limit. The only exception is for figures-only pages, which are not counted.
- Graphs are undirected and simple (i.e., no loops, no parallel edges) by default.
- In case you were able to solve one of these problems without having to use anything you learnt in this course, that's okay. I have deliberately included one "CS 25" level problem.
- You may refer to both textbooks and to all of the material linked from the course web site. You may refer to math textbooks or the web site `mathworld.wolfram.com` to look up any math you may have forgotten. But referring to anything else (other web sites, other published papers, etc.) is a violation of the honor code.
- You may not discuss this exam in any detail whatsoever with anyone, even after you have submitted it. If you have questions on the course's material, please ask me; I am willing to help you to any extent with understanding of the material. However, please don't ask me if your approach to a problem is correct; I cannot answer that.
- You may find the exam hard. This is expected. Don't worry and good luck!

Section	Points	Score
1	15	
2	10	
3	20	
4	20	
5	15	
6	20	
Total	**100**	

Final Exam
Due March 14, 2005; 5:00pm

Prof. Amit Chakrabarti
Computer Science
Dartmouth College

1. Suppose M and N are finite sets with $|M| = m$, $|N| = n$ and $m \geq n$. A hash function $h : M \to N$ is said to be *b-perfect* on the subset $S \subseteq M$ if it maps at most b keys to any one slot; formally:

   $$\forall\, i \in N \,:\, |\{x \in S : h(x) = i\}| \;\leq\; b\,.$$

   A family $\mathcal{H}$ of such hash functions is said to be a *b-perfect hash family* if, for every $S \subseteq M$ with $|S| = n$, there exists a function $h \in \mathcal{H}$ such that h is b-perfect on S.

   Prove that there exists an $O(\log m)$-perfect hash family $\mathcal{H}$ with $|\mathcal{H}| \leq m$.

   [15 points]

2. Before splay trees were invented, a *move-to-root heuristic* was being considered for self-adjusting binary search trees. The idea was that after a FIND operation, the last node searched in the tree would be moved to the root by repeatedly rotating the edge between it and its parent. In splay tree terminology, that node would be moved to the root by repeatedly using ZIG operations, rather than the complicated combination of ZIG-ZIG, ZIG-ZAG and ZIG used in splay trees.

   Demonstrate that the move-to-root heuristic works poorly on certain access sequences. More precisely, show that starting with any n-node binary search tree that is adjusted using this heuristic, one can perform a sequence of m FIND operations that require $\Omega(mn)$ time in total.

   Your operations should only try to find keys that are actually in the tree. Since this is a big-Ω bound, you may assume that m and n are sufficiently large.

   [10 points]

3. Recall that a cut in a graph is a partition of its vertex set into two nonempty parts, say X and $\overline{X}$. The cut is said to be a (unweighted) min-cut if the *number* of edges crossing from X to $\overline{X}$ is minimized. Note that an n-vertex graph has exactly $2^{n-1} - 1$ distinct cuts (not $2^n - 2$, because $(X, \overline{X})$ and $(\overline{X}, X)$ are considered to be the same cut). However, in a *connected* graph, not too many of these cuts can be min-cuts.

   Find an upper bound, in terms of n, on the number of distinct min-cuts in a connected n-vertex graph. Prove that your bound is tight by giving a family of examples that covers every $n \geq 2$.

   [20 points]

4. A string that reads the same backwards as forwards is called a palindrome; examples are `RADAR`, `110010011`, `X`, `MALAYALAM`,[1] and `SAIPPUAKAUPPIAS`.[2]

   4.1. Given an input string $x \in \Sigma^*$, for some finite alphabet Σ, the *longest palindromic prefix* (LPP) problem asks for the longest prefix y of x such that y is a palindrome. Describe an algorithm for this problem that runs in time $O(n)$, where $n = |x|$.

   [10 points]

   4.2. Given $x \in \Sigma^*$, the *longest palindromic subsequence* (LPS) problem asks for any longest subsequence y of x such that y is a palindrome. Describe an algorithm for this problem that runs in time $O(n^2)$, where $n = |x|$.

   [10 points]

[1] A language spoken in southern India.
[2] Finnish for "soap salesman."

To get credit you must analyze the running times of your algorithms and prove their correctness. Note that the running times should be independent of $|\Sigma|$.

Some sample inputs and outputs:

Input	LPP Output	Possible LPS Output
ABACAA	ABA	AACAA
ABACXYCXZABCB	ABA	BACXCAB
AWKWARDNESS	AWKWA	AWKWA
MAGICAL	M	AGA

5. Given an n-vertex edge-weighted graph G and a list $\langle t_1, \ldots, t_k \rangle$ of k distinguished vertices (known as *terminals*) of G, a *multiway cut* of G with respect to the terminals is a partition of the vertex set into k parts such that each part contains exactly one terminal. The weight of such a multiway cut is the sum of the weights of all edges which cross from one part to another. The *minimum multiway cut* problem asks for a multiway cut of smallest possible weight; all edge-weights are assumed to be positive.

   For $k = 2$ this is the familiar s-t min-cut problem, which as you know can be solved in polynomial time. However, for $k \geq 3$, the problem is NP-hard. However, an approximation algorithm can be designed as follows. Let G_i denote the graph obtained by starting with G and merging all the terminals *except for* t_i into a single supervertex u_i. For each $i \in \{1, 2, \ldots, k\}$, find a t_i-u_i min-cut of G_i. Suitably "combine" these min-cuts into a multiway cut of G.

   Fill in the details to obtain a 2-approximation algorithm for the minimum multiway cut problem. Then improve this to a $(2 - \frac{2}{k})$-approximation algorithm by observing that one of the k min-cuts is "unnecessary." To get any credit, you must prove that your algorithm correctly finds a multiway cut and prove the approximation ratios.

   [15 points]

6. This problem asks you to improve upon the 2-approximation we gave in class for the metric travelling salesman problem (TSP).

   A perfect matching in a graph is one that includes every vertex. If the graph has edge costs, the cost of a matching is defined to be the sum of the costs of the edges in the matching.

   > FACT 1: Given a complete graph with edge costs and with an even number of vertices, a minimum cost perfect matching can be found in polynomial time.
   >
   > FACT 2: If all vertices of a multigraph (i.e., parallel edges allowed, but no loops) have even degree then the multigraph has an Eulerian tour, i.e., a tour that traverses each *edge* exactly once. Such an Eulerian tour can be found in polynomial time.

   Using the two facts above, give a polynomial time $(3/2)$-approximation algorithm for metric TSP. The MST-based 2-approximation algorithm we studied in class might be a useful starting point.

   [20 points]