# Design and Analysis of Algorithms
# End-Semester Examination

Time: 3 Hours | November 28, 2012 | Marks: 80

**Instructions:**

1. Solve as many questions as you can.
2. Try to optimize your algorithm as much as possible.
3. Whenever you are asked to design an algorithm, write a correctness proof, and analyze the time complexity.

**Questions:**

1. A network is given as an undirected simple graph $G = (V, E)$. Each edge $e = (u, v)$ has an associated failure probability $p_e$, which is the probability that the link connecting nodes $u$ and $v$ fails. Assuming that these probabilities are independent, give an efficient algorithm to find a path between two given vertices $s$ and $t$ which has the least failure probability. (Note that a path fails if any link on that path fails.) **5**

2. There are $n$ balls in a row. Each ball is colored *red*, *white*, or *blue*. You are required to sort these balls so that all the red balls precede all the white balls, which in turn precede all the blue balls. The only operations you are allowed to perform are *check*, in which you can look at the color of a ball, and *swap* in which you can swap the positions of two balls. Give an $O(n)$ time algorithm for this problem. **7**

3. Consider the problem of sorting a sequence of $n$ integers with many duplications, such that there are only $O(\log n)$ many distinct elements. Give an $O(n \log \log n)$ time algorithm to sort such a sequence. Why doesn't this violate the $\Omega(n \log n)$ lower bound on sorting? **8**

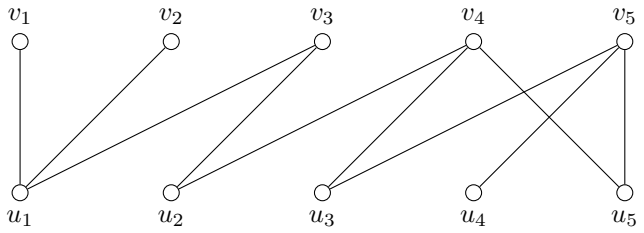
4. Consider the following method of performing a walk on a binary search tree. Find the minimum element in the tree and then find its $n - 1$ successors. In other words, if the minimum is $a$, then first find $a$, then find $b =$ successor of $a$, then $c =$successor of $b$ and so on. Prove that this can be implemented in $\Theta(n)$ time. (Assume that the tree is given using pointers, and that there are *left, right, parent* pointers in each node. You should not assume that the tree is balanced. If you assume so, you will get only 50% credit.) **8**

5. Let $G = (U, V, E)$ be a bipartite graph. Suppose we form a flow network from $G$ in the usual manner, i.e. add vertices $s$, $t$ with edges from $s$ to all $u \in U$ and from all $v \in V$ to $t$, with all edge capacities 1, and all edges oriented from $s$ to $U$ to $V$ to $t$.

   (a) Show that there exists a *min-cut* $(A, B)$ such that $s \in A$, $t \in B$ and there are no edges leaving $A$ and entering $V \setminus A$. By $V \setminus A$ we mean the set obtained by removing $V \cap A$ from $V$. **7**

   (b) Using the min-cut as found above, find a vertex cover for $G$ of size same as the min-cut. **4**

(c) Show $(A, B)$ and the vertex cover for the given graph: **4**




(d) Using the min-cut obtained in (a) above, prove the following version of Hall's theorem: Let $G = (U, V, E)$ be a bipartite graph with $|U| = |V| = n$. There is a *perfect matching* in $G$ if and only if, for each $A \subseteq U$, $|N(A)| \geq |A|$. Here $N(A)$ is the set of neighbours of $A$. A matching $M$ is said to be *perfect* if every vertex of $G$ is matched in $M$. **4**

6. The *Dominating Set* problem is stated as follows: The input is an undirected graph $G = (V, E)$. The goal is to find a subset $V' \subseteq V$ of minimum cardinality such that every vertex in $V \setminus V'$ has an edge to some vertex in $V'$.

   In the *Bipartite Dominating Set* problem, the input is a bipartite graph $G = (X, Y, E)$ and the goal is to find a subset $X' \subseteq X$ of minimum cardinality such that every vertex in $Y$ is adjacent to at least one vertex in $X'$.

   (a) Prove that the bipartite dominating set problem is NP-complete . **4**

   (b) Show that the bipartite dominating set problem reduces to dominating set problem. **4**

   (c) Express the dominating set problem (optimization version) as an integer linear program (No proof needed). **3**

   (d) Suppose the graph $G$ is $d$-regular, i.e. each vertex has exactly $d$ neighbours. Consider the following algorithm for this case:

   Relax the integrality constraints of the above integer linear program to get an LP. Compute an optimal solution to this LP. If a variable has value at least 1/(d+1), replace it by 1.

      i. Prove that the algorithm described above gives a valid dominating set for $G$ (not necessarily minimum). **3**

      ii. Prove that the dominating set obtained according to the above algorithm is a $(d + 1)$-approximation to the optimum dominating set. **4**

7. An instance of the *graph coloring problem* consists of an undirected graph $G = (V, E)$ and an integer $k$, which denotes the number of colors. The question is whether there exists an assignment of colors to vertices so that the two end-points of each edge get different colors. (In an assignment, each vertex is assigned one of the $k$ colors.)

   (a) It is easy to see that, given $G$ and $k$, there may not be any coloring which satisfies the above condition. Give such an example for $k > 2$. **3**

   (b) Given $G$ and $k$, let us consider the following optimization version of the coloring problem: Call an edge *satisfied* if its two end-points have different colors. The goal is to find an assignment of colors to vertices of $G$ so as to maximize the number of satisfied edges. Show that it is always possible to satisfy at least $\frac{k-1}{k}|E|$ edges. **6**

   (c) Show how to find one such coloring in polynomial time. **6**