Collaboration is allowed, but you must write your own solutions.

**Problem 1**

*Note: In (a), SEARCH need not run in logarithmic time. In (b), define an appropriate potential function so that insertion runs in amortized $O(1)$ time. In (c), one does not expect the implementation to be very efficient.*

**Problem 2**

*Note: Assume that $\alpha$ is strictly larger than $\frac{1}{2}$ (and strictly less than 1). Ignore deletions (as everything would be similar to insertions).*

**Problem 3**

**Problem 4**
Solve: [Kleinberg Tardos] Chapter 3, problem 12, page 112.

*Hint: Create a directed graph with two nodes for each person, one representing his birth-date and the other representing his death-date.*

**Problem 5**
Consider a flow network on four nodes $\{s, t, a, b\}$ and five edges with capacities

$$c(s,a) = c(s,b) = c(a,t) = c(b,t) = 1000 \quad \text{and} \quad c(a,b) = 1.$$ 

Show that starting with the zero flow, the Ford-Fulkerson max-flow algorithm could take upto 2000 augmentation steps.
Reading assignment: If edge capacities are irrational numbers, Ford-Fulkerson algorithm may never terminate and the flow could converge to a value other than the value of a max-flow. Study the example on page 2 of: http://www.cs.uiuc.edu/class/sp07/cs473g/lectures/14-maxflowalgs.pdf

Problem 6

As suggested by Dinitz as well as Edmonds and Karp, during the execution of the Ford-Fulkerson max-flow algorithm, suppose we always select an augmenting path in the residual graph that contains a minimum number of edges. Prove that the Ford-Fulkerson algorithm terminates in \( O(mn) \) iterations where \( n \) and \( m \) are the number of vertices and edges in the network respectively.

Hint: Consider the BFS starting at the source \( s \) and the corresponding layers in the residual graph \( R \). As we know, every edge in \( R \) either stays in the same layer, goes to a previous layer or goes to a layer that is immediately next. Let \( R' \) be the residual graph after an augmentation step using a shortest \( s \)-\( t \) path. How does \( R' \) look compared to \( R \)? Let \( \text{dist}(s,t) \) denote the length of a shortest \( s \)-\( t \) path. Show that \( \text{dist}(s,t) \) in \( R' \) is at least equal to \( \text{dist}(s,t) \) in \( R \). Show that this distance must increase by at least one after at most \( m \) iterations.

Problem 7

In an undirected graph \( G(V,E) \), a vertex cover is a set \( S \subseteq V \) such that every edge \( e \in E \) has at least one endpoint in \( S \). A matching is a set \( M \subseteq E \) such that no two edges in \( M \) share an endpoint.

1. Show that if \( S \) is a vertex cover and \( M \) is a matching, then \( |M| \leq |S| \).

2. Give an example of a graph where the maximum size of any matching is strictly less than the minimum size of any vertex cover (Hint: Look at graphs with three vertices).

3. Now assume that the graph is bipartite, i.e. the vertex set is partitioned as \( V = U \cup W \) and every edge \( e \in E \) has one endpoint in \( U \) and the other in \( W \). The goal is to prove that the maximum size of a matching is equal to the minimum size of a vertex cover. Prove this by designing a polynomial time algorithm to simultaneously find a maximum matching as well as a minimum vertex cover.

Hint: Construct a flow network as described in class by adding a source \( s \) and a sink \( t \). Compute the max-flow and the corresponding min-cut. Let the min-cut be \( \{s\} \cup A \cup B \) and \( \{t\} \cup (U \setminus A) \cup (W \setminus B) \). Let \( C \subseteq (W \setminus B) \) be the set of neighbors of \( A \) in \( W \setminus B \). Show that \( (U \setminus A) \cup B \cup C \) is a vertex cover. Why is its size equal to the value of the max-flow (which in turn equals the size of maximum matching)?
Problem 8 (Randomized Algorithms)
Solve: [Kleinberg Tardos] Chapter 13, problem 1, page 782.

Problem 9 (Randomized Algorithms)

Hint: Consider an agent such that there are $k$ agents with a higher bid than her. What is the probability that her bid results in an update of $b^*$?

Problem 10 (Randomized Algorithms)

Problem 11
Given a directed graph $G(V, E)$, call two vertices $u$ and $v$ equivalent if there is a path from $u$ to $v$ and vice versa. Give a polynomial time algorithm (there are $O(|V| + |E|)$ time implementations but don’t worry in this problem) to partition the vertex set $V$ into non-empty sets $P_1, P_2, \ldots, P_k$ such that:

- For every $1 \leq i \leq k$, all vertices in $P_i$ are equivalent to each other.
- For every $1 \leq i < j \leq k$, no vertex in $P_i$ is equivalent to any vertex in $P_j$.

Such a partition is called decomposition into strongly connected components. Now construct a directed graph $G'$ on vertex set $\{1, 2, \ldots, k\}$ and add an edge $(i, j)$ if there is an edge from some vertex in $P_i$ to some vertex in $P_j$. Show that $G'$ is acyclic.

Problem 12 (Do not submit)
Solve all problems in this assignment from Fall’06 (highly recommended as a preparation for the final exam):
http://www.cs.nyu.edu/courses/fall06/G22.3520-001/ps5.pdf