CS 473: Algorithms, Fall 2009

HBS 5

This HBS contains review problems for midterm 1. We don't expect you to solve all of them during the hbs. You can use the starred problems to prepare for the exam and you can skip them during the hbs.

Try to solve 3 of the first 6 problems during the hbs. You can skip the starred parts of these problems. *Note that the problems are ordered by topic, not difficulty.*

Problem 1. Give a tight asymptotic bound for the following recurrences.

- **Warmup:**
  1. $A(n) = A(n-1) + 1$, where $A(0) = 0$
  2. $B(n) = B(n-5) + 2$, where $B(0) = 17$
  3. ★ $C(n) = C(n-1) + n^2$, where $C(0) = 0$
  4. $D(n) = 3D(n/2) + n^2$, where $D(1) = 1$
  5. ★ $E(n) = 4E(n/2) + n^2$, where $E(1) = 1$
  6. ★ $F(n) = 5F(n/2) + n^2$, where $F(1) = 1$
- **Real Practice:**
  1. $A(n) = A(5n/8) + A(3n/8) + n$, where $A(n) = 1$ if $n < 8$
  2. ★ $B(n) = B(n/3) + 3B(n/5) + B(n/15) + n$, where $B(n) = 1$ if $n < 15$
  3. ★ $C(n) = 2C(n-1) + C(n-2)$, where $C(0) = 1, C(1) = 2$

Problem 2. Let A be an array consisting of n integers. An integer is a *majority* element if it appears more than $\lfloor n/2 \rfloor$ times in A. You are given such an array, but you are not allowed to compare two numbers directly. Instead, you are given access to an equality tester which, given two numbers, returns true if the two numbers are equal and false otherwise. *In other words, the only operation you can do with the numbers is to pick up any two of them and plug them into the equality tester.* Give an algorithm that decides whether A has a majority element using only $O(n \log n)$ invocations of the equality tester.

Problem 3. A *data stream* is an extremely long sequence of items that you can only read once, in order. A good example of a data stream is the sequence of packets that pass through a router. Data stream algorithms must process each item in the stream quickly, using as little memory as possible; there is simply too much data to store, and it arrives too quickly for any complex computations. Every data stream algorithm looks roughly like this:

```
DoSomethingInteresting(stream S)
  repeat
      x ← next item in S
      ⟨⟨do something fast with x⟩⟩
  until S ends
  return ⟨⟨something⟩⟩
```

You are given a stream S consisting of n integers and an integer k and you want to find the k smallest elements in the stream.

(a) Give an algorithm for the problem that runs in $O(n \log k)$ time and uses $O(k)$ space.

(b) ★ Give an algorithm for the problem that runs in $O(n)$ time and uses $O(k)$ space. [*Hint: Store at most* $2k$ *of the integers. When there are* $2k$ *integers already stored and you need to make room for an additional integer, use the* SELECTION *algorithm to decide which integers to throw away.*]

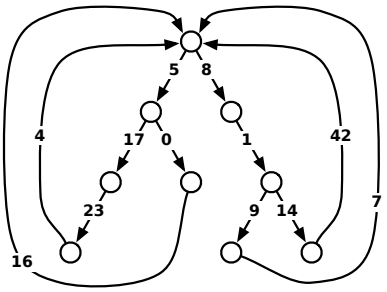
Problem 4. Let G be a connected graph and let v be a vertex in G. Show that if T is both a DFS tree and a BFS tree rooted at v, then $G = T$.

Problem 5. Let $G = (V, E)$ be a directed graph in which each node u has a label k_u, where k_u is an integer. For each node u, we want to find the largest label of a node v that can reach u.

(a) Solve the problem assuming G is a DAG.

(b) Solve the problem by extending to general graphs via the strongly connected components meta-graph.

Problem 6. A looped tree is a weighted, directed graph built from a binary tree by adding an edge from every leaf back to the root. Every edge has a non-negative weight.

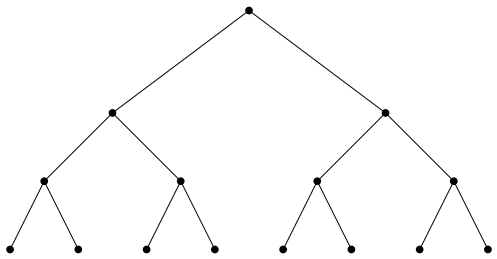


(a) How much time would Dijkstra's algorithm require to compute the shortest path between two vertices u and v in a looped tree with n nodes?

(b) Give a faster algorithm.

You can skip the remaining problems during the hbs. You can use them (and the previous problems) to practice for the exam.

Problem 7★. A binary tree is a rooted tree in which every node has at most 2 children. A node is called a leaf if it has no children. Non-leaf nodes are called internal nodes. A *complete* binary tree is a binary tree in which every internal node has exactly 2 children and all the leaves are on the same level.



A complete binary tree with 15 nodes.

Consider a complete binary tree T on n nodes. Each node v of T is labeled with an integer x_v (the node labels are distinct). A node v of T is a *local minimum* if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge. You are given such a tree T, but the labeling is specified in the following implicit way: for each node v, you can determine the value x_v by probing the node v. Give an algorithm that finds a local minimum of T using only $O(\log n)$ probes to the nodes of T.

Problem 8*. Let G be an undirected graph with n nodes. Suppose that G contains two nodes s and t, such that every path from s to t contains more than $n/2$ edges.

(a) Prove that G must contain a vertex v that lies on *every* path from s to t.

(b) Describe an algorithm that finds such a vertex v in linear time.