CPS 230 Final

Fall 2000

7pm-10pm, Thursday December 14
Closed book exam

25 points **Problem 1**

In this problem we are given n elements and want to find the k smallest elements in sorted order. Design an optimal (comparison-based) algorithm for the problem. Prove that your algorithm is optimal in the comparison model of computation.

25 points **Problem 2**

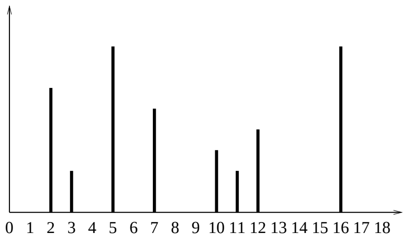
In this problem we consider a monotonously decreasing function $f : N \to Z$ (that is, a function defined on the natural numbers taking integer values, such that $f(i) > f(i+1)$). Assuming we can evaluate f at any i in constant time, we want to find $n = \min\{i \in N | f(i) \leq 0\}$.

We can obviously solve the problem in $O(n)$ time. Describe a more efficient algorithm.

35 points **Problem 3**

In this problem we consider a data structure $\mathcal{D}$ for maintaining *bar-diagrams*, where we have a *bar* of height $h \geq 0$ associated with every non-negative integer.

The following is an example of a bar-diagram:



We want to support the following operations:

- Init($\mathcal{D}$): Create a bar-diagram where all bars have height 0.
- Change($\mathcal{D}$,b, k): Add k to bar b (k can be negative but a bar cannot have negative height).
- Height($\mathcal{D}$,b): Return the height of bar b.
- TotalSum($\mathcal{D}$): Return the sum of the height of all bars.

a) Describe an implementation of $\mathcal{D}$ such that Init and TotalSum runs in $O(1)$ time and such that Change and Height runs in $O(\log n)$ time. Here n is the number of non-zero bars in the diagram.

Next we consider the problem of extending the data structure with an operation IntervalSum as follows:

- IntervalSum($\mathcal{D}$,b_1, b_2): Return the sum of the height of all bars b such that $b_1 \leq b \leq b_2$.

b) Describe an implementation of $\mathcal{D}$ such that Init and TotalSum run in $O(1)$ time and all other operations—including IntervalSum—run in $O(\log n)$ time.

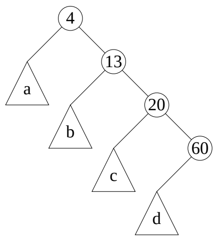
40 points **Problem 4**

In this problem we consider binary trees containing elements in heap order. We say that a node in the tree is *heavy* if there is at least as many elements in its left subtree as in its right subtree. Otherwise we call it *light*. We refer to the path from the root to the rightmost leaf as the *rightmost path* and to the path to the leftmost leaf as the *leftmost path*.

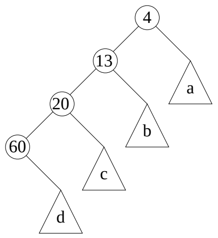
a) Prove that if the number of heavy nodes on the rightmost path is t then the tree contains at least $2^t - 1$ nodes (note that if the tree is of size n this implies that the rightmost path contains $O(\log n)$ heavy nodes).

We implement an INSERT(e) operation as follows: First we exchange the left and right subtrees of all nodes on the rightmost path and then we insert e in the appropriate position in the resulting leftmost path.

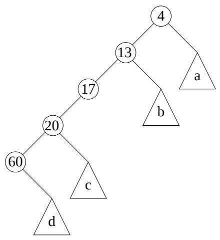
Example: Consider performing INSERT(17) on the following tree:



After exchanging the left and right subtrees on the path from 4 to 60 we obtain the following tree:



And after inserting 17 we end up with the following tree:



b) Which nodes change status from *heavy* to *light* or from *light* to *heavy* when performing INSERT(e)? Justify your answer.

c) Show that the amortized time for an INSERT(e) in a sequence of n insertions on an initially empty tree is $O(\log n)$.

35 points **Problem 5**

A *palindrome* is a string that reads the same from front and back. Any string can be viewed as a sequence of palindromes if we allow a palindrome to consist of one letter.

> Example: "bobseesanna" can e.g be viewed as being made up of palindromes in the following ways:
> "bobseesanna" = "bob" + "sees" + "anna"
> "bobseesanna" = "bob" + "s" + "ee" + "s" + "anna"
> "bobseesanna" = "b" + "o" + "b" + "sees" + "a" + "n" + "n" + "a"

We are interested in computing $\textsc{MinPal}(s)$ defined as the minimum number of palindromes from which one can construct s (that is, the minimum k such that s can be written as $w_1 w_2 \dots w_k$ where $w_1, w_2, \dots, w_k$ are all palindromes).

> Example: $\textsc{MinPal}$("bobseesanna")=3 since "bobseesanna" = "bob" + "sees" + "anna" and we cannot write "bobseesanna" with less than 4 palindromes.

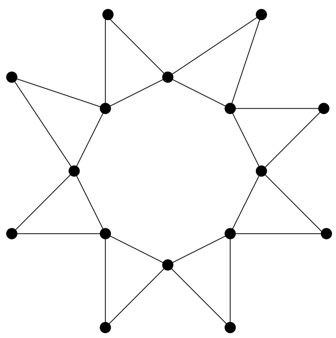
It is easy to see that $\textsc{MinPal}(s[i,j]) = \min_{i \le k < j}\{\textsc{MinPal}(s[i,k]) + \textsc{MinPal}(s[k+1,j])\}$

a) Show that the straightforward implementation of $\textsc{MinPal}(s)$ based on the above formula results in an exponential time algorithm.

b) Describe and analyze a better algorithm.

40 points **Problem 6**

A *star-graph* is an undirected graph of the following form—it's Christmas isn't it? :-)



More precisely, a star-graph consists of a center cycle of length $n/2$ where each pair of connected vertices are also both connected to a third vertex. We assume that each edge has a positive integer weight and that the graph is given in the normal edge-list representation without any extra information.

a) How long would it take for Prim's or Kruskal's algorithms to find the minimal spanning tree of a star-graph?

b) Describe and analyze a more efficient algorithm for finding the minimal spanning tree of a star-graph. Remember to prove that your algorithm is correct.