# CPS 230 Midterm

**Fall 2000**

2:20pm-3.35pm, Friday October 13th
Closed book exam

*25 points* **Problem 1**

a) Show that $\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$

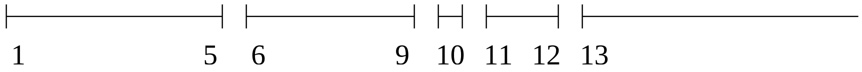
b) Solve the recurrence

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n-1) + n(n+1)(n-1) & \text{if } n \geq 2 \end{cases}$$

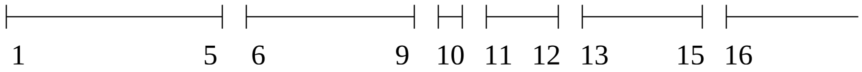
*30 points* **Problem 2**

In this problem we consider the so-called INTERVAL-UNION-SPLIT-FIND problem. We want to maintain a set of disjoint intervals covering all the natural numbers (one interval is considered to be infinite) such that we can support the operations UNION, SPLIT, and FIND. UNION combines two consecutive intervals, SPLIT splits an interval in two, and FIND returns a unique representative for a given interval.

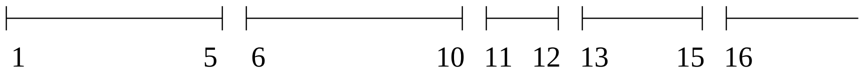
Example: Consider the following intervals:




Performing a SPLIT on the last interval at 16 results in the following set of intervals:




if we then perform a UNION on the interval $[6, \ldots, 9]$ and the interval after that $([10, 10])$ we obtain the following set of intervals:




and if we then perform a FIND on element 14 we get the unique representative for the interval $[13, \ldots, 15]$. It is natural to choose the first element in an interval as the unique representative. Thus FIND on element 14 would return 13.

The INTERVAL-UNION-SPLIT-FIND can now be formalized as maintaining a data structure $\mathcal{D}$ under the following operations:

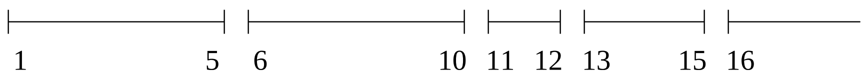
- INIT($\mathcal{D}$): Create an interval-union-split-find structure containing the interval $[1, \ldots, \infty]$.
- FIND($\mathcal{D}$,$x$): Return the representative (the first element) in the interval containing $x$.
- UNION($\mathcal{D}$,$x$): Union the interval containing $x$ with the interval after that. No UNION is performed if $x$ is in the last interval.
- SPLIT($\mathcal{D}$,$x$): Split the interval $[a, \ldots, b]$ containing $x$ into two intervals $[a, \ldots x-1]$ and $[x, \ldots, b]$. No SPLIT is performed if $x$ is the representative of $[a, \ldots, b]$, that is, if $x = a$.

a) Describe an implementation of $\mathcal{D}$ such that INIT runs in $O(1)$ time and such that all other operations run in $O(\log n)$ time, where $n$ is the number of intervals in the structure.

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

- FINDNEXT($\mathcal{D}$,$x$,$d$): Return the representative for the first interval after the interval containing $x$ of length at least $d$ (if existing).

Example: Performing a FINDNEXT($\mathcal{D}$,7,3) on the intervals




returns 13.

b) Describe an implementation of $\mathcal{D}$ such that INIT runs in $O(1)$ time and such that all other operations—including FINDNEXT—run in $O(\log n)$ time, where $n$ is the number of intervals in the structure.

*25 points* **Problem 3**

Recall that in a worst-case efficient $O(n \log n)$ time version of quick-sort, we use an $O(n)$ time *partition* algorithm to partition an array of $n$ elements into two equal sized arrays such that all elements in one of the arrays are smaller than all elements in the other array (implemented using linear-time median finding).

a) Describe an $O(n \log k)$ time *k-partitioning* algorithm that partitions an array of $n$ elements into $k$ equal sized arrays $A_1, A_2, \ldots, A_k$ such that the elements in array $A_i$ are smaller than the elements in array $A_{i+1}$ for all $1 \leq i < k$. You can assume that $k$ is a power of two (that is, that $k = 2^i$ for some integer $i$).

b) A "smart" UNC Chapel-Hill professor claims that he has discovered an $O(n)$ time $\sqrt{n}$-partition algorithm (in the comparison model). Prove that he is wrong.

*20 points* **Problem 4**

In this problem we consider an array $A$ of length $n$ for which we know that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. We say that $A[x]$ is a *local minimum* if $A[x-1] \geq A[x]$ and $A[x] \leq A[x+1]$. Note that $A$ must have at least one local minimum.

We can obviously find a local minimum in $A$ in $O(n)$ time. Describe a more efficient algorithm for finding a local minimum.