Divide and Conquer Algorithms (part 2 of CSC 282),

http://www.cs.rochester.edu/~stefanko/Teaching/11CS282

1 Schedule

"Homework" problem sessions are in CSB 601, 6:15-7:15pm on Sep. 14 (Wednesday) and on Sep. 21 (Wednesday); held by Rintaro Kuroiwa and Josh Pawlicki.

Homework is **due Sep. 22** (Thursday).

"Exam" problem session is in CSB 601, 4:45-5:45pm on Sep. 26 (Monday).

EXAM #2 will be on **Tuesday, Sep. 27**.

2 List of algorithms covered in the class

(B-basic, I-intermediate, A-advanced):

I: Karatsuba-Offman (p.47, DSV).
B: Mergesort (p.50, DSV).
I: Linear time deterministic selection (p.189, CLRS).
A: Linear expected-time selection (p.53, DSV).
A: FFT (p.57, DSV).

Important concepts, problems, theorems, and algorithms:

- Recurrences, master theorem.
- Merge-sort, linear-time select algorithm (median).
- Evaluation, interpolation, convolution.

3 Basic Homework - solve and turn in

2.1 Find the unique polynomial of degree 2 such that $p(-1) = 2$, $p(0) = 3$, and $p(1) = 6$.

2.2 Solve the following recurrence relations and give O bound for each of them (you can use Master theorem).

- $T(n) = 5T(n/4) + n$.
- $T(n) = 2T(n/3) + 1$.
- $T(n) = 9T(n/3) + n^2$.
- $T(n) = 7T(n/7) + n$.

2.3 Compute the convolution of $(1, 0, 0, 1, 0, 1)$ with $(0, 1, 0, 0, 1)$. (DEFINITION: convolution of two sequences $a_0, \ldots, a_n$ and $b_0, \ldots, b_m$ is a sequence $c_0, \ldots, c_{m+n}$, where

$$c_k = \sum_{i=\max(0,k-m)}^{\min(k,n)} a_i b_{k-i},$$

i. e., $c_0, \ldots, c_{m+n}$ is the coefficient sequence of the polynomial $(a_0 + a_1 x + \cdots + a_n x^n)(b_0 + b_1 x + \cdots + b_m x^m))$.

2.4 The convolution of sequence $(1, 2, 3, 4)$ with an unknown sequence S is sequence $(5, 11, 17, 23, 4)$. Compute the sequence S.

2.5 Let A and B be two sets. Their sum $A + B$ is defined to be

$$A + B := \{a + b \,|\, a \in A,\ b \in B\},$$

for example if $A = \{1, 4, 6\}$ and $B = \{2, 5\}$ then $A + B = \{3, 6, 8, 9, 11\}$. Give $O(n \log n)$ algorithm whose input is two sets $A, B \subseteq \{1, \ldots, n\}$ and the output is set $A + B$. (HINT: use the FFT algorithm.)

2.6 (**due Sep. 22**) We are given k sorted lists $A_1, \ldots, A_k$. The total number of elements in the lists is n. We would like to merge the lists into one sorted list B (the number of elements in B will be n). Give an algorithm which solves this problem in time $O(n \log k)$.

2.7 (**due Sep. 22**) We are given two arrays of integers $A[1..n]$ and $B[1..n]$, and a number X. Design an algorithm which decides whether there exist $i, j \in \{1, \ldots, n\}$ such that $A[i] + B[j] = X$. Your algorithm should run in time $O(n \log n)$.

4 Advanced Homework - solve and turn in

In problems 2.8, 2.9, 2.10, **do NOT use hashing** (in practice hashing would be a great technique for these problems even though it does not have worst-case guarantees), **do NOT use linear sorting** (the assumptions of the problems do not allow for a linear sorting algorithm, such as, radix-sort).

2.8 (**due Sep. 22**) We are given an array A with n elements and a number C. Assume that the sum of the elements in A is larger than C. We would like to compute the size of the smallest subset of A whose elements sum to at least C. (For example, if $A = [8, 3, 9, 2, 7, 1, 5]$ and $C = 18$ then the answer is 3; the set is $\{7, 8, 9\}$.) Give a linear-time algorithm for this problem. (HINT: use the linear-time SELECT algorithm)

2.9 (**due Sep. 22**) We are given an array of integers $A[1..n]$. We would like to determine whether there exists an integer x which occurs in A more than $n/2$ times (i. e., whether A has a majority element). Give an algorithm which runs in time $O(n)$. (HINT: use the linear-time SELECT algorithm.)

Example: For $A = [3, 1, 2]$ the answer is NO. For $A = [3, 1, 3]$ the answer is YES.

2.10 (**due Sep. 22**) We are given an array of integers $A[1..n]$. We would like to determine whether there exists an integer x which occurs in A more than $n/3$ times. Give an algorithm which runs in time $O(n)$. (HINT: use the linear-time SELECT algorithm)

2.11 (**due Sep. 22**) We are given an array of integers $A[1..n]$ which is almost sorted in the following sense: for all $i \in \{1, \ldots, n - k\}$ we have $A[i] \leq A[i + k]$. Give an algorithm which sorts the array A. Your algorithm should run in time $O(n \log k)$.

2.12 (**due Sep. 22**) We are given two sorted arrays A and B, each containing n elements. Assume that the arrays do not contain duplicates, and the elements in A are different from elements in B. We would like to compute the median of $A \cup B$. For example if $A = [1, 2, 3, 4, 5]$ and $B = [6, 7, 8, 9, 10]$ then the median is 5; if $A = [1, 3, 5, 7, 9]$ and $B = [2, 4, 6, 8, 10]$ then the median is again 5. Give an $O(\log n)$ algorithm for this problem.

2.13 (**due Sep. 22**) We are given an array of integers $A[0..n-1]$. Dr. Median uses the following program

```
for i from 0 to n − 1 do
    for j from 0 to n − 1 do
        C[i+n*j] = A[i]+A[j];
return the median of C[0], C[1], ..., C[n^2 − 1].
```

Dr. Median needs a faster program for the task. Give an $O(n \log n)$ algorithm that returns the same answer as the algorithm above.

5 Additional problems from the book (do not turn in)

Try to solve the following problems. A few of them will be on the quiz. We will go over the ones that you choose in the problem sessions.

- 2.4, 2.5, 2.8, 2.10, 2.12, 2.16, 2.17, 2.18, 2.21, 2.26, 2.27, 2.28, 2.29, 2.30, 2.31, 2.34.