

## 1 Schedule

Homework is **due Sep 28**.  
The **QUIZ** will be on **Thursday, Sep. 30**.

## 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).

## 3 Basic material

**Important concepts, problems, theorems, and algorithms:**

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

**Example problems (solve, but do NOT 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.

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

**2.3** Let

$$A[1..25] = [7, 2, 23, 4, 9, 5, 1, 24, 3, 10, 6, 8, 17, 12, 14, 19, 25, 22, 15, 11, 13, 20, 16, 21, 18].$$

Suppose that we use the deterministic linear-time **select** algorithm on  $A$ . Compute the pivot used by the algorithm to split  $A$ .

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

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

i. e.,  $c_0, \dots, c_{m+n}$  is the coefficient sequence of the polynomial  $(a_0 + a_1x + \dots + a_nx^n)(b_0 + b_1x + \dots + b_mx^m)$ ).

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

**2.6** Evaluate the polynomial  $x^3 + 3x^2 - x + 1$  on the 4-th roots of unity. (DEFINITION:  $n$ -th roots of unity are  $\exp(2\pi ik/n)$ , where  $k = 0, \dots, n - 1$ . For example, 4-th roots of unity are:  $i, -1, -i, 1$ .)

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

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

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  $A, B \subseteq \{1, \dots, n\}$  and the output is set  $A + B$ . (HINT: use the FFT algorithm.)

## 4 Additional homework

In problems 2.8, 2.11, 2.12, **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 28)** 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 28)** We are given  $k$  sorted lists  $A_1, \dots, A_k$ . 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  $O(n \log k)$ .

**2.10 (due Sep 28)** 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.

**2.11 (due Sep 28)** 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.

**2.12 (due Sep 28)** 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)

**2.13 (due Sep 28)** We are given an array of integers  $A[1..n]$  which is almost sorted in the following sense: for all  $i \in \{1, \dots, n - k\}$  we have  $A[i] \leq A[i + k]$ . Give an algorithm which sorts the array  $A$ . Your algorithm should run in time  $O(n \log k)$ .

**2.14 (due Sep 28)** 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  $i, j \in \{1, \dots, n\}$  such that  $A[i] + B[j] = X$ . Your algorithm should run in time  $O(n \log n)$ .

## 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.