Review for Test 2

Dynamic programming. There will be a question where you are given a problem that can be solved using dynamic programming. You will be asked to give a recursive characterization of the problem, a thorough and clear explanation of what the tables mean, and a verbal description of a strategy for populating the tables.

Greedy algorithms. There will be a problem that can be solved greedily. You will be asked to write an algorithm to solve this problem and prove that the problem has the greedy choice.

Polynomials and the FFT. There will be a problem that involves representing polynomials, operating on them, and/or transforming polynomials from one representation to another.

Finite automata and push-down atomata. There will be a series of factual questions about DFAs, NFAs, regular expressions, context-free languages, and PDAs. Know the definitions of the various machines and the theorems from the sections we covered in class (you will not need to reproduce the definitions and theorems, but the questions will require you to understand their significance). In particular, understand machine equivalence and the equivalence between classes of machines.

15-4. Let L_i be the length of the ith word and, as the problem states, M be the maximum number of characters per line. Let $C[i]$ be the total penalty (cost) of the best way to print the words from i to the end, with i beginning a new line. Then the recursive characterization is

$$C[i] = \begin{cases} 0 & \text{if } n - i - 1 + \sum_{k=i}^{n-1} L_k \leq M \\ \min_{j \mid i<j\leq x} C[j] + \left(j - i - 1 + \sum_{k=i}^{j-1} L_k\right)^3 & \end{cases}$$

. . . where x is the greatest index such that $x - i - 1 + \sum_{k=i}^{j-1} L_k \leq M$.

16-4. The greedy choice is to schedule the task with shortest completion time first.

Proof. *Suppose S is a set of tasks and A be an ordering of the tasks in S that minimizes their total completion time (which, it should be noted, also minimizes the average completion time). Let $a_1, a_2, \ldots$ refer to the tasks in S in the order specified by A, that is A indicates that a_1 should be scheduled first, a_2 second, etc. Let the indices on processing time p_i correspond to this ordering, that is p_1 is the processing time of the tasked scheduled first by A.*

Let a_k be the task with minimal processing time, that is for all $i \in [1, n], p_k \leq p_i$. Now consider an ordering of the tasks like A but with task a_k moved to the front.

Now consider the difference between the total completion time of A and the total completion time of the new ordering: the completion times of tasks a_i where $i > k$ are unchanged; the completion time of task a_k is decreased by $\sum_{i=1}^{k-1} p_i$; the completion time of each task a_i where $i < k$ is increased by p_k. Thus the difference is

$$\begin{array}{rcll} (k-1)p_k - \sum_{i=1}^{k-1} p_i & \geq & (k-1)p_k - sum_{i=1}^{k-1} p_k & \text{because } p_k \leq p_i \\ & = & (k-1)p_k - (k-1)p_k & \\ & = & 0 & \end{array}$$

Since the difference is greater than or equal to 0, the new ordering has total processing time less than or equal to the total processing time of the supposed minimal ordering. Therefore it is safe to put the task with least processing time first. $\square$

Code (yes, this is it):

```
public static void scheduleBatch(BatchProcessor.Task[] tasks) {
    Arrays.sort(tasks, new Comparator<BatchProcessor.Task>() {
        public int compare(BatchProcessor.Task o1, BatchProcessor.Task o2) {
            return o1.processingTime - o2.processingTime;
        }
    });
}
```

For part **b.** the greedy choice is to schedule, at any point, the with least remaining non-zero completion time, of those task that have been released.

30-1.a

$$(a_0 + a_1 x) \cdot (b_0 + b_1 x) = \underbrace{a_1 b_1}_{c_2=\alpha} x^2 + \underbrace{(a_0 b_1 + a_1 b_0)}_{c_1=\gamma-\alpha-\beta} x + \underbrace{a_0 b_0}_{c_0=\beta}$$

$$\underbrace{(a_0 + a_1) \cdot (b_0 + b_1)}_{\text{hint in book},\gamma} = a_0 b_0 + a_0 b_1 + a_1 b_0 + a_1 b_1$$

$$\underbrace{(a_0 b_1 + a_1 b_0)}_{c_1=\gamma-\alpha-\beta} = (a_0 + a_1) \cdot (b_0 + b_1) - \underbrace{a_0 b_0}_{c_0=\beta} - \underbrace{a_1 b_1}_{c_2=\alpha}$$

30-1.b.i

$$\sum_{i=0}^{n-1} a_i x^i \cdot \sum_{i=0}^{n-1} b_i x^i = \underbrace{\sum_{i=0}^{\frac{n}{2}-1} a_i x^i \cdot \sum_{i=0}^{\frac{n}{2}-1} b_i x^i}_{\alpha} + x^n \underbrace{\sum_{i=0}^{\frac{n}{2}-1} a_{i+\frac{n}{2}} x^i \cdot \sum_{i=0}^{\frac{n}{2}-1} b_{i+\frac{n}{2}} x^i}_{\beta}$$

$$+ x^{\frac{n}{2}} \left(\sum_{i=0}^{\frac{n}{2}-1} (a_i + a_{i+\frac{n}{2}}) x^i \cdot \sum_{i=0}^{\frac{n}{2}-1} (b_i + b_{i+\frac{n}{2}}) x^i - \alpha - \beta \right)$$

The recurrence is

$$T(n) = 3T\left(\frac{n}{2}\right) + f(n)$$

where $f(n)$ is dominated by the cost of additions and subtraction in the list comprehensions, which are linear. Applying the Master method, $a = 3$, $b = 2$, and $f(n) = \Theta(n)$. Note that $\lg 3 \approx 1.585$. Thus $f(n) = O(n^{\lg 3 - \epsilon})$ for $\epsilon = \frac{1}{2}$, for example. Therefore, by the Master method, $T(n) = \Theta(n^{\lg 3})$.

30-1.b.ii

$$\sum_{i=0}^{n-1} a_i x^i \cdot \sum_{i=0}^{n-1} b_i x^i = \underbrace{\sum_{i=0}^{\frac{n}{2}-1} a_{2i} x^{2i} \cdot \sum_{i=0}^{\frac{n}{2}-1} b_{2i} x^{2i}}_{\alpha} + x^2 \underbrace{\sum_{i=0}^{\frac{n}{2}-1} a_{2i+1} x^{2i} \cdot \sum_{i=0}^{\frac{n}{2}-1} b_{2i+1} x^{2i}}_{\beta}$$

$$+ x \left(\sum_{i=0}^{\frac{n}{2}-1} (a_i + a_{i+1}) x^i \cdot \sum_{i=0}^{\frac{n}{2}-1} (b_i + b_{i+1}) x^i - \alpha - \beta \right)$$

The analysis is the same.