CSE 541T Advanced Algorithms January 25, 2019

Monkey Business: Dynamic Programming Practice

You may have heard the statement, "If a million monkeys sat at a million typewriters for a million years, they still couldn't write Shakespeare." Primate Publishing, Inc. has hired you to test this hypothesis, in hopes of finding a new, low-cost way of producing best-sellers.

The basic idea is as follows. Let W be the text of Shakespeare's complete works, consisting of an ordered list of words $w_1 \dots w_n$. The company's army of trained monkeys bangs out another list of n words $V = \{v_1 \dots v_n\}$. Your job is to compare the texts W and V word-by-word and determine whether any passage of V is substantially similar to the corresponding passage (of the same length) in W.

To score the similarity in any interval $[i..j]$, you simply compare corresponding words of passages $[w_i \dots w_j]$ and $[v_i \dots v_j]$, accounting a score of $+1$ for each match and $-\alpha$ for each mismatch (for some constant $\alpha > 0$). The score of the interval is the sum of these per-word scores. For example, the following interval has two mismatches in eight words and hence scores $6 - 2\alpha$:

W : But	soft!	What	light	through	yonder	window	breaks?
V : But	soft!	What	*banana*	through	yonder	*monkey*	breaks?

Note: if every nonempty interval has a negative score, the best choice is the empty interval, which has score 0.

Problem: Given W and V as above, find an interval $[i^*..j^*]$ (of any length) whose score is maximal among all subintervals in $[1..n]$.

Find an efficient dynamic-programming algorithm for this problem. Your algorithm should require worst-case time $\Theta(n)$.

1 Small Group Work

For this part, you will divide into groups of 3-4 (depending on number of attendees). Each group should work through the exercise below collaboratively. *Write down* your solution – you may be asked to present it to another group or to everyone present.

1. Consider the following initial choice set: "Either the interval contains text position n, or it does not." It's trivial to argue the Complete Choice Property for this set. Do both choices leave a subproblem of the original? Why or why not?

2. Prove the Inductive Structure and Optimal Substructure Properties for this initial choice set.

3. Construct a recurrence based on this initial choice set, propose an evaluation order for it, and prove the complexity of your algorithm.

4. Does your algorithm achieve the required time $\Theta(n)$? If not, can you find a way to speed it up? *Hint*: develop a separate recurrence to compute the solutions for each subproblem's non-recursive case in total time $\Theta(n)$.

5. Your employers are only interested in *long* passages, i.e. high-scoring intervals of length at least k words, for some fixed $k > 1$. Can you extend your algorithm to find the best-scoring interval of length $\geq k$, while still running in time $\Theta(n + k)$ (*not* $\Theta(nk)$)?

You can use the dynamic programming handout as a template to help you write the proof.

2 Shared Critique

Your TA will organized this part of the exercise. Be prepared to explain and defend your algorithms and proofs.