Algorithms in the Time of COVID19

Midterm+DP[1] *- Recitation*[2] *8*

November 6, 2020

[1] Contact: Liangzu Peng (`lp2528@nyu.edu`).

[2] Hand-written notes available at: `https://www.dropbox.com/sh/x1z104c22d51pox/AACiJdDSKe2SDZw3qNljNApka?dl=0`

Midterm

Problem 1. Given f and g, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$). Briefly explain your answer.

1. $f(n) = \log(n^4 - 1), g(n) = \log(n^2 + 3n)$.
2. $f(n) = \log(2^{n^2} + n^2), g(n) = \log(n^{472})$.

Midterm

Problem 2. Consider the recurrence
$T(n) = T(n/4) + T(n/2) + n$ with $T(1) = 1$.

1. Using a recursion tree, determine a tight asymptotic upper bound on $T(n)$.
2. Prove your upper bound using induction.

Problem 3. Suppose you have some procedure FASTERMERGE that given two sorted lists of length m each, merges them into one sorted list using m^c steps for some constant $c > 0$. Write a recursive algorithm using FASTERMERGE to sort a list of length n and also calculate the run-time of this algorithm as a function of c. For what values of c does the algorithms perform better than $O(n \log n)$?

Midterm

Problem 4. Let A be an array with n *distinct* integer elements in sorted order. Consider the following algorithm IdFind(A, j, k) that finds an $i \in \{j, \ldots, k\}$ such that $A[i] = i$, or returns FALSE if no such element i exists.

IdFind(A, j, k)

- if $j > k$ return FALSE
- Set $i := \ldots$
- if $A[i] = \ldots$ return $\ldots$
- if $A[i] < \ldots$ return IdFind$(A, \ldots, \ldots)$
- return IdFind$(A, \ldots, \ldots)$

1. Fill in the blanks (denoted $\ldots$) to complete the above algorithm.
2. Prove correctness and analyze the running time of the algorithm.

Problem 5. Given symbols A, B, C, D, E, F with frequencies $f_A = 0.1$, $f_B = 0.1$, $f_C = 0.2$, $f_D = 0.3$, $f_E = 0.15$, $f_F = 0.15$.

1. Draw the decoding tree of the optimal prefix code for this set of symbols.
2. What is the average encoding length of this set of symbols?
3. **Bonus [5 marks]:** What is the optimal ternary tree and corresponding encoding length for this set of symbols? (In a ternary tree, each node has *at most* 3 children. A ternary tree corresponds to a prefix code over $\{0, 1, 2\}$. For a given set of symbols, the optimal ternary tree gives the minimal average encoding length of all symbols.)
4. **Bonus [5 marks]:** Describe an algorithm for finding the optimal ternary tree when given a set S of n symbols with frequencies $f_1, \ldots, f_n$.

Problem 6. Prove the correctness of the following algorithm that finds the optimal ternary tree.

Exercises - P4.4 in the problem set

We say that an array A is *c-nice* for a *constant* c if for all $1 \leq i < j \leq n$ such that $j - i \geq c$, we have that $A[i] \leq A[j]$. For example, a 1-nice array is completely sorted (in ascending order). In this problem we will sort such c-nice arrays A using InsertionSort and QuickSort and compare the results.

Exercises - P4.4 in the problem set

Problem 7. What is the worst case running time of insertion sort on a c-nice array?

Exercises - P4.4 in the problem set

Consider now a run of Quicksort on a c-nice array (where the pivot element is chosen (deterministically) as the last element of the array).

Problem 8. (a) Derive a lower bound on the rank q of the pivot.[3] (b) Argue that after partitioning, the two subarrays $A[1...q-1]$ and $A[q+1...n]$ to the left and to the right of the pivote, respectively, are both c-nice. (c) Let $T(n)$ be the best-case running time of quicksort on a c-nice array with n elements. Derive a reccurence for $T(n)$ and solve it.

[3] Among n elements, the i-th smallest element has rank i.

Dynamic programming

Dynamic programming - weighted interval scheduling

- Input: n intervals $[s_1, f_1], \ldots, [s_n, f_n]$ with weights $w_1, \ldots, w_n$.
- Output: a set S of disjoint intervals that maximize $\sum_{i \in S} w_i$

Dynamic programming - weighted interval scheduling

next[i]: first non-overlapping interval on the right of i
(:=n+1 if not exist)

Opt(i): maximum income that we can earn using intervals from i to n

$$\underbrace{Opt(1)}_{\text{what we want}} = \max\{\underbrace{Opt(next[1])+w_1}_{\text{choose 1}},\ \underbrace{Opt(2)}_{\text{not choose 1}}\}$$

Dynamic programming - weighted interval scheduling

Algorithm (top-down weighted interval scheduling)

- sort the intervals by non-decreasing starting time, i.e. $s_1 \le s_2 \le \dots \le s_n$.
- compute next[i] for $1 \le i \le n$. — O(n log n)
- Compute opt (1). — O(n)

opt(i) // recursive function

if $i = n+1$, return 0.

opt(i) = max { w_i + opt(next[i]), opt(i+1) }. // choose i, or not choose i

Question. How to compute next[i] in $O(n \log n)$ time?

Dynamic programming - Knapsack

- Input: n items, each with weight w_i and value v_i, and a positive integer C.
- Output: a set S with $\sum_{i \in S} w_i \leq C$ maximizes $\sum_{i \in S} v_i$.

Problem 9. Assume that there are n objects, and the i-th object weights $w_i > 0$ kilograms and has value v_i. Here n and v_i are both positive integers with $n > v_i$. With a knapsack of capacity $W > 0$ kilograms, design an algorithm to find the maximum total value of the objects with which we can fill the knapsack.