CSc 345 — Analysis of Discrete Structures
Fall 2024 (McCann)

http://cs.arizona.edu/classes/cs345/fall24/

Homework #3

(95 points)

Due Date: Thursday, November 7^{th}, 2024, at the beginning of class

Write complete, legible answers to each of the following questions. (questions of the form "#.#" are exercises from the Shaffer PDF text). Show your work, when appropriate, for possible partial credit. **This is not a group project; do your own work!** *If you need help, remember that the TAs and I have office hours for just this eventuality, and you may also post general questions on Piazza.*

On the due date, by the start of class, submit a PDF containing your answers on gradescope.com. Be sure to assign pages to problems after you upload your PDF. (Need help? See "Submitting an Assignment" on https://help.gradescope.com/.) If you need to submit your solutions within the 24–hour late window, Gradescope will be configured to accept them. Solutions submitted after 3:30pm on the 8th will not be accepted. Want to be safe and submit your homework early? Please do! You can resubmit an updated PDF before the due date, if necessary.

1. [35 points] Graphs. (Shaffer Ch. 11, pp. 399-401.)

   (a) [5] 11.10 (use Bellman–Ford's algorithm instead of Dijkstra's, and you can use either the book's table or the one I used in class)

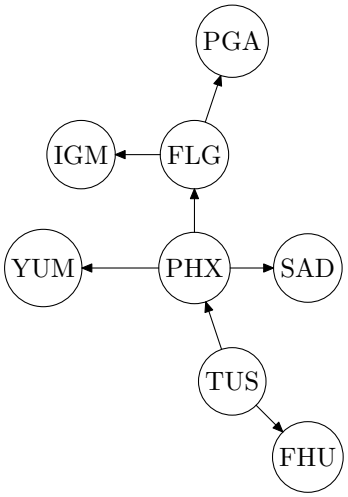
   (b) [5] 11.17

   (c) [5] 11.18

   (d) [10] 11.23 (note that there are two questions; explain your answers to both, and also note that "MST" means Minimal–cost Spanning Tree – I used "MCST" in class)

   (e) [5] Determine two correct topological sort orderings of the digraph shown below.

   (f) [5] Imagine that the edge (IGM,PHX) is added to the graph below. Can the resulting graph be topologically sorted? Why or why not?



(Continued...)

2. [45 points] Internal Sorting. (Shaffer Ch. 7, pp. 257-61.)

   (a) [10] Using weak induction, prove that the Insertion Sort algorithm, as presented in class, will produce a sorted list of data elements.

   (b) [10] 7.6 (drop Heapsort and BinSort, add Counting Sort)

   (c) [5] 7.16 (a,b)

   (d) [20 total] Typically, Internal Merge Sort is presented as doing two–way merging – that is, two sorted runs are merged. Consider the following pseudocode for three–way Internal Merge Sort, where `list` is the array of data elements, and `low` and `high` are the low and high indices of the range of data elements to be sorted:

```
1  3WIMS(list,low,high):
2      if low < high:
3          f <-- last index of the first third of list
4          s <-- last index of the second third of list
5          3WIMS(list,low,f)
6          3WIMS(list,f+1,s)
7          3WIMS(list,s+1,high)
8          3WayMerge(list,low,f,s,high)
```

   i. (5) Determine the recurrence relation $K(n)$ and initial condition $K(1)$ that describes the quantity of key comparisons performed by `3WIMS`, and construct at least two additional equivalent recurrence relations for $K(n)$ using the "Find the Pattern" (a.k.a. Backward Substitutions) approach. Assume that the key comparison cost of `3WayMerge()` is $2n$, where n is the initial quantity of data items in the list.

   ii. (5) Using the three (or more) versions of the $K(n)$ recurrence relations you created in part (a), create the "in general" version of the $K(n)$ recurrence relation and solve that version, producing the closed–form (non–recurrence) expression for $K(n)$.

   iii. (10) Using weak induction, prove that your closed–form expression for $K(n)$ from part (b) is the solution to the recurrence relation from part (a). We want to see a complete, clear, easy–to–follow proof — the basis step, the inductive step (starting with a clear "if–then" that states the inductive hypothesis and the conjecture being proven, followed by a proof of that conjecture that clearly shows where the inductive hypothesis is being used).

   I did two "find the pattern" examples in class on Sept. 19th. See also Question 6 on Exam 1 and the online "Problems for Practice: Recurrence Relations" handout.

3. [10 points] External Sorting. Consider this sequence of 50 two–letter pairs:

```
EL DO BY ED EM EF UT ZA EN WO ER MU KI NO MI OD FE AG AD HA
HE TA RE SI PE SO AB HI ID AA GO AI KA AS IT AL PA UN TI PI
QI NU LA MA LI MO BI BE OH ON
```

   Use Multiway External Merge Sort to sort the pairs into ascending order. Assume that your buffers can hold up to four two-letter pairs each, and you have four buffers available. Show your collection of sorted runs at the end of each pass of the algorithm. Also, during passes 1-n, every **fifth** time you write out the content of the output buffer (that is, after the 5th, 10th, 15th, etc. writes), show the content of the input buffers at that time (the output buffer, having just be written, will be empty). That's how we'll be able to tell if you're following the algorithm correctly. (Yes, we know that this is a pain. But it's also a good exercise to make sure that you know how the algorithm works, and that's the reason to do homeworks.)

4. [5 points] Interpolation Search. You have an array of one million elements, indexed 0 through 999,999. The range of element values is 0 .. 499,999,000,000. (That is, array[0] = 0 and array[999,999] = 499,999,000,000.) The search target is 8,675,309.

   Use Interpolation Search as described in class to perform the first three probes into the data. Assume that the element found at probe index i is equal to $\lfloor \frac{i^2}{2} \rfloor$. What are the three indices of the three probes? Show your calculation expressions for all three probes for full credit; just listing the indices will be worth only one point.