CMPT 307 — Data Structures and Algorithms

Exercises on Stable Matchings and Sorting. Due: Friday, September 27th (through Crowdmark)

Reminder: the work you submit must be your own. Any collaboration and consulting outside resourses must be explicitely mentioned on your submission.

1. Describe a $\Theta(n \log n)$ time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose difference is exactly x.

2. Suppose that in the MergeSort algorithm the input sequence split into k subsequences ($k > 2$, but fixed). Determine the running time of such algorithm. Does it have any advantages comparing to the standard MergeSort? (You may wish to look into Chapter 4 of CLR.)

3. We explore the issue of truthfulness in the Stable Matching problem and specifically in the Gale-Shapley algorithm. The basic question is: Can a man or a woman end up better off lying about his or her preferences? More concretely, we suppose each participant has a true preference order. Consider a woman w. Suppose w prefers man m to m', but both m and m' are low on her list of preferences. Can it be the case that by switching the order of m and m' on her list of preferences (i.e. by falsely claiming that she prefers m' to m) and running the algorithm with this false preference list, w will end up with a man m'' that she truly prefers to both m and m', but would not end up with using the true list of preferences?

   Resolve this question by doing one of the following two things:

   (a) Give a proof that, for any set of preference lists, switching the order of a pair on the list cannot improve a woman's partner in the Gale-Shapley algorithm; or

   (b) Give an example of a set of preference lists for which there is a switch that would improve the partner of a woman who switched preferences.

4. An $m \times n$ Young tableau is an $m \times n$ matrix such that the entries of each row are in sorted order from left to right and the entries of each column are in sorted order from top to bottom. Some of the entries of a Young tableau may be ∞, which we treat as nonexistent element. Thus a Young tableau can be used to hold $r \leq mn$ finite numbers.

   (a) Show how to insert a new element into a non-full $m \times n$ Young tableau in $O(m+n)$ time.

   (b) Give a $O(m+n)$ time algorithm to determine whether a given number is stored in a given $O(m+n)$ Young tableau.

5. Show that the worst-case running time of Heapify-Up on a heap of size n is $\Omega(\log n)$.

6. (a) What is the running time of Quicksort whe all elements of array A have the same value?
   (b) Show that the running time of Quicksort is $\Theta(n^2)$ when the array A contains distinct elements and is ordered in decreasing order.

7. The Quicksort algorithm contains two recursive calls to itself. After the call to Partition, the left subarray is recursively sorted and then the right subarray is recursively sorted. The second recursive call in Quicksort is not really necessary; it can be avoided by using an

iterative control structure. This technique, called *tail recursion*, is provided automatically by good compilers. Consider the following version of Quicksort, which simulates tail recursion.

```
Quicksort'(A,p,q)
while p<r do
     Partition and sort left subarray
     set q:=Partition(A,p,r)
     Quicksort'(A,p,q−1)
     set p:=q+1
endwhile
```

(a) Argue that Quicksort'(A,1,length(A)) correctly sorts the array A

Compilers usually execute recursive procedures by using a stack that contains pertinent information, including the parameters values, for each recursive call. The information for the most recent call is at the top of the stack, and the information for the initial call is at the bottom. When a procedure is invoked, its information is *pushed* onto the stack; when it terminates, its information is *popped.* Since we assume that array parameters are represented by pointers, the information for each procedure call on the stack requires $O(1)$ stack space. The *stack depth* is the maximum amount of stack space used at any time during a computation.

(b) Describe a scenario in which the stack depth of Quicksort' is $\Theta(n)$ on an n-element input array.

(c) Modify the code of Quicksort' so that the worst-case stack depth is $\Theta(\log n)$.

8. Show that there is no comparison sort algorithm whose running time is linear for at least half of the $n!$ possible inputs. What about a fraction of $1/n$ of the inputs of length n? What about a fraction $1/2^n$?