FUNDAMENTAL ALGORITHMS
OLD MIDTERM SOLUTIONS

1. (20) Give an algorithm HORSE with the following property. The input
is two arrays $A[1 \cdots N]$, $B[1 \cdots N]$, both arrays in increasing order.
The output is an array $C[1 \cdots (2N)]$ which has all the values of the
arrays $A, B$ and is in increasing order. How long does your algorithm
take. (Brief reason please!)
Solution: This is the MERGE algorithm as given in class.

2. (20) Illustrate the operation of COUNTINGSORT on the array

$A = (2, 1, 2, 1, 0, 0, 0, 1)$

with $n = 8$ and $k = 2$. Pictures and some well chosen words, please.
(You do not need every detailed step but you must make clear the
main steps.)
Solution: You have an auxiliary array $C[0 \cdots 2]$ which is initially zero.
For $i = 1$ to $n = 8$, running through $A$, you increment $C[A[i]]$ by one.
This gives a count on $C$: $(3, 3, 2)$. Now for $j = 1$ to $2$ ($k$) you set
$C[j] \leftarrow C[j] + C[j-1]$. This turns $C$ into a cumulative count: $(3, 6, 8)$.
Finally for $i = 8$ down to $1$ (going down the array $A$) you set (say)
value $= A[i]$ and then position $= C[value]$. Then you place the value
$A[i]$ in that position in $B$: $B[position] \leftarrow value$. Critically, you then
decrement $C[value]$ so that next time you hit that value it will be
going to a different place.

3. (20) For the following algorithms let $T(N)$ denote the total number of
times the step after the WHILE step is reached. For the first algorithm
give (five points) an exact formula for $T(N)$. For the second algorithm
first (ten points) give $T(N)$ as a precise sum. Then (five points) Find
$T(N)$ is the form $T(N) = \Theta(g(N))$ for a standard $g(N)$. Reasons
please!

(a) X=1
WHILE X < N
do X=2*X

Solution: After applying the doubling $t$ times you will have $X = 2^t$. So $T[N]$ is one more than the the maximal $t$ with $2^t < N$
which is one more than the maximal $t$ with $2^t \leq N - 1$ which
is the maximal $t$ with $t \leq \lg(N - 1)$ which is $1 + \lfloor \lg(N - 1) \rfloor$.

(April full credit if you missed the $-1$.)

(b) FOR I=1 TO N
    X=1
    WHILE $X^2 \leq I$
        $X++$

Solution: For each $I$ you will apply the $X++$ step when $X \leq \sqrt{I}$
so $[\sqrt{I}]$ times. Thus

$$T(N) = \sum_{I=1}^{N} [\sqrt{I}]$$

Without the floor the splitting in half methods makes $T(N) = \Theta(N^{3/2})$.
The floors only effect each term by at most one and therefore the sum by at most $N$ which is negligible compared to
$\Theta(N^{3/2})$ so the answer is $T(N) = \Theta(N^{3/2})$.

4. (10) In hashing, what are collisions? Describe one method (your
choice!) for dealing with them.

Solution: Collisions are when there are two items $x, y$ which have the
same hash value, that is, $h(x) = h(y)$. There are two methods: One is
to have the hash table be a table of linked lists and simply add $y$ to
the linked list. The other is to have a probe sequence, if $h(y)$ fails you
have backups $h_1(y), h_2(y), \ldots$ that you try.

5. (20) Let $A$ is a max-heap with heapsize $N$. Describe a program called
here BIGGULP$(A, i, key)$ that replaces $A[i]$ by a value $key$ which is
bigger than $A[i]$ and then restores the heap property. How long does
BIGGULP take? How long does BIGGULP take in the special case when
$i = 1$?

Solution: Set $y = i$ and reset $A[y] \leftarrow key$. Now while $y \neq 1$
you check whether $A[parent[y]] < A[y]$. If it is you interchange $A[parent[y]]$ and
$A[y]$ and reset $y$ to $parent[y]$. Else you stop. When $i = 1$ you stop
immediately so it takes 1 (or $O(1)$) steps. (Its bad form to say it takes
zero steps because, after all, you have to check that $i = 1$.)

6. (15) You want to sort five elements $a, b, c, d, e$ using seven paired
comparisons. Assume that your question is “Is $a < b$” and that the answer
was Yes. Assume that your second question is “Is $a < c$.” Using the
Information-Theoretic Lower Bound prove that you will not be able
to sort the elements.

**Solution:** Suppose the answer is Yes. (For a method to work it has to work in all cases.) Now we know that \( a \) is the smallest of \( a, b, c \). Of the 120 original orderings, one third of them, or 40, are still valid. But we only have five further questions and \( 40 > 2^5 \) so we cannot succeed by the Information Theoretic Lower Bound.

7. (15) There is an algorithm \texttt{RABBIT(A,B)} that multiplies two \( n \times n \) matrices \( A, B \) by performing seven multiplications of \( (n/2) \times (n/2) \) matrices and then performing \( O(n^2) \) further operations. Create a recursive equation for the time \( T(n) \) that \texttt{RABBIT(A,B)} takes and use the Master Theorem to give \( T(n) \) asymptotically.

**Solution:** This is Strassen’s Algorithm. The recursion is

\[
T(n) = 7T(n/2) + O(n^2)
\]

which is the low overhead case of the Master Theorem so that \( T(n) = \Theta(n^\alpha) \) where \( \alpha = \log_2 7 \).

8. (15) Let \( A[1\cdots N] \) be an array with all entries integers between 0 and \( N \). How long would \texttt{RADIX-SORT} take to sort \( A \) assuming that we use base 2 (that is, binary)? (Assume the entries \( A[I] \) are already given as binary strings in the input.) You must give an argument for your answer.

**Solution:** Each counting sort would take time \( O(N) \). But there are \( \lg N \) binary digits so the total time is \( O(N \lg N) \).

9. (5) State the binary-search-tree property. (That is, the condition that the keys are required to fulfill.)

**Solution:** For any node \( x \) and any node \( y \) in the left subtree of \( x \), the value at \( y \) is \( \leq \) the value at \( x \). For any node \( x \) and any node \( y \) in the right subtree of \( x \), the value at \( y \) is \( \geq \) the value at \( x \).