Midterm Exam solutions

1 Short answers

[40 points]

Part A. Solve the following recurrence relation to within \( \Theta \) accuracy.

\[
T(n) = \begin{cases} 
8T(n/2) + 11n^3 & n \geq 2 \\
n & n < 2.
\end{cases}
\]

Use the formula, \( a = 8 = b^3 \), so \( \Theta(n^3 \log n) \)

Part B. Suppose we run the closest point algorithm on \( n = 2^k \) points. What is the minimum number of pairs of points we do a distance computation on (where a “distance computation” computes the Euclidean distance between two points)? Note that you are being asked to analyze the best case input. Briefly justify your answer.

\( n/2 \) since we end with \( n/2 \) base problems of size 2, each of which takes 1 compare. In the combine step no points are in the strip in the best case, so no compares.

Part C. Suppose we have an undirected graph \( G = (V,E) \) with \( n \) nodes and each node in \( G \) has at most 5 neighbors.

i) How long does it take to find the connected components in \( G \) if \( G \) is represented using an adjacency list structure?

ii) Same question, but now \( G \) is stored in an adjacency matrix.

Give your answer in terms of \( n \) and in big Theta notation.

In both cases we use Breadth First Search to find the CC. In case i) we can do it in \( \Theta(n + m) = \Theta(n) \) since \( m \leq 5n \).

In case ii) it takes \( \Theta(n^2) \) to scan the matrix.

Part D. Solve the following recurrence relation to within \( \Theta \) accuracy. Justify your answer.

\[
T(n) = \begin{cases} 
T(n/2) + T(n/3) + n^2 & n \geq 2 \\
n & n < 2.
\end{cases}
\]

\( T(n) = \Theta(n^2) \)
We answer is at least $n^2$, the direct work. We can upper bound this by

$$T(n) = T(n/2) + T(n/2) + n^2 \quad n \geq 2$$

and use the formula to show this is $\Theta(n^2)$

**Part E.** Consider the proposal algorithm for the stable marriage problem with $n$ men and women. If all men rank the same woman (say woman $a$) as their last choice:

How many proposals does woman $a$ get?

Justify your answer.

Woman $a$ gets one proposal. As we argued in class, when a man proposes to his last choice woman, all others are paired, so she accepts and the algorithm ends.

2 Coin Changing

Suppose we had coins of value 1, 10, 25 and 100 cents.

**Part A.** Does the greedy algorithm (use 100 cent coins till below 100, then 25, 10 and 1’s) work here? Give a counter example or proof to justify your answer.

No, as before 30 cents should be 3 dimes, not $Q +$ five pennies.

**Part B.** What is the maximum number of quarters in an optimal solution? Briefly justify your answer.

3. If 4 replace by a dollar coin. Need 3 for 75 cents.

**Part C.** What is the maximum number of dimes in an optimal solution? Briefly justify your answer.

4. If five, replace by two quarters. Need 4 for 40 cents.

**Part D.** Describe an efficient algorithm to compute $Change(N)$, the smallest number of coins required to make change of $N$. Your goal is an algorithm which is as fast as possible for large $N$.

It follows from B.C that the change excluding dollars is always less than two dollars (3 Q's, 4D, 4P). Thus we can use the greedy algorithm until less than 2 dollars, then find best of the rest.

So we first compute optimal change for all values up to 99 cents (using the dynamic programming approach discussed in class) where we compute $Change(n)$, for $n = 1$ to 25 by brute force (or greedy), then use $Change(n) = 1 + \min \{Change(n-1), Change(n-10), Change(n-25)\}$ for $n = 1, 99$.

For $n = 100, 199$ we use

$Change(n) = 1 + \min \{Change(n-1), Change(n-10), Change(n-25), Change(n-100)\}$

Once we have these values, if $N \leq 199$ we just look it up. If $N \geq 200$ we compute $R = N \mod 100$, and $M = N - R - 100$.

So we start by using $M/100$ dollar coins, and then look up the remaining change for $100+R$. 

Part E. what is the running time of your algorithm of part D?
O(1) since we use constant preprocessing (to compute the initial Change values) and then a constant number of arithmetic operations.

3 Greedy [16 points]

Consider the activity selection problem where we have a single lecture hall and n activities with start times $s_i$ and finish time $f_i$ for the $i$th activity. Our goal is still to schedule the maximum number of activities.

Part A. As before, suppose that we sort the finish times so $f_1 \leq f_2 \leq \cdots \leq f_n$, but now we start by scheduling activity $n$ (with the latest finish time). Is activity $n$ always part of some optimal solution? Justify your answer by giving a counter-example to justify "no", or a reason why it always works if "yes".

NO. Consider an activity which starts at time zero and also has the latest finish time. Thus we get no others.

Part B. Suppose we have many ties among the finish times (so there are only $m << n$ distinct finish times). Thus we propose to now solve the problem by first identifying the $m$ distinct finish times, and we then sort these to get $f_1 < f_2 < \cdots < f_m$.

We then select one activity for each finish time, and run the greedy algorithm on these $m$ activities.

i) Describe a good way to find these $m$ distinct finish times (so your solutions should have a fast expected run time).

ii) What is the overall expected run time of this new solution?

i) We use hashing to find the distinct finish times (and select the representative activity). This takes expected $O(n)$ time to hash all $n$ finish times using chaining and a size $n$ hash table (we hash the finish times, and throw away any later activities which have the same finish time; thus our table has one of each unique time).

We now sort the activities in the table by finish, and apply the usual greedy algorithm.

ii) This takes ($\Theta(n + m \log n)$ expected time.

4 Dynamic Programming [24 points]

We can generalize the Shuffle Problem to take four strings $X = x_1, \ldots, x_n$, $Y = y_1, \ldots, y_n$, $W = w_1, \ldots, w_n$ and $Z = z_1, \ldots, z_n$.

Our goal is determine if $Z$ is a shuffle of $X$, $Y$ and $Z$. Thus, for example, for the strings $X=\text{AAC}$ and $Y=\text{CAA}$ and $W=\text{ACB}$, $Z=\text{ACACACAB}$ is a shuffle of the three strings (taking the symbols from $\text{XXYYWWXYWX}$ in that order), while $Z=\text{ABACABAC}$ is not a shuffle since you can’t get a $B$ before any $C$’s are used. Let $X_i$, $Y_i$, $W_i$, $Z_i$ represent the first $i$ characters of strings $X, Y, W, Z$.

Let $A[i,j,k]$ be 1 if it is possible to shuffle $X_i$, $Y_j$, and $W_k$ to get $Z_{i+j+k}$

$A[i,j,k] = 0$ otherwise.
**Part A.** Describe how to compute $A[1, 1, k]$ for $k = 1, 2, \ldots, n$.

We need to match $x_1, y_1,$ and $W_k$ to $Z_{k+2}$ we can do this by brute force by trying each pair of places which matches $x_1, y_1$ and then try to fill in the rest with $W_k$. Alternately, we can first compute, $A[1, 1, 1]$ by trying the six orderings of $x_1, y_1, w_1$ to see if they match $Z_3$, then compute the $A[1, 0, k]$ and $A[0, 1, k]$ values by trying single positions for $x_1$ in $Z_{k+1}$, then similarly for $y_1$ and use the formula of part b.

**Part B.** Write a formula for computing $A[i, j, k]$ when $i, j$ and $k$ are all greater than 1. Which values of $A$ do you need in order to compute this new value?

\[ A[i, j, k] = 1 \text{ if } (A[i-1, j, k] = 1 \text{ and } x_i = Z_{i+j+k}) \text{ OR } (A[i, j-1, k] = 1 \text{ and } y_j = Z_{i+j+k}) \text{ OR } (A[i, j, k-1] = 1 \text{ and } w_k = Z_{i+j+k}) \]

Otherwise its zero (since last character must match end of one of the 3 strings, and then the rest shuffle).

**Part C.** What is the running time of your algorithm? How much space does it use?

We have a 3D array each side is $n$, and we fill in each in $O(1)$ so $O(n^3)$ time and space.

**Part D.** If you only want a yes-no answer (yes if $Z$ is a shuffle, no if its not), describe how to reduce the space used.

We only need to keep two $n^2$ size planes: previous one and next one.