CMPS 6610 Extra Credit Answers

In this extra credit assignment, we will test and review concepts you have learned since the midterm exam. Please add your written answers to `answers.md` which you can convert to a PDF using `convert.sh`. Alternatively, you may scan and upload written answers to a file named `answers.pdf`.

1. **Algorithmic Paradigms**

If I had to choose a favorite algorithm from this unit, I would pick Kruskal's algorithm. I don't particularly love any algorithm, but Kruskal's feels the most intuitive to me. When I do Minimum Spanning Tree problems by hand, Prim's algorithm can feel mentally overwhelming because I have to keep track of all the edges from all the nodes already included in the tree. In contrast, Kruskal feels more sequential. I simply sort the edges by weight and add the smallest one as long as it doesn't form a cycle. That step-by-step process feels clearer and less stressful to follow manually, which is why Kruskal's is the one I prefer.

2. **Divide and Conquer**

Let's assume we are given a set of points and we want to find the closest pair.

If we use divide and conquer:

1. split the set into two
2. recursively find the closest pair

This would not be an optimal solution because what if the closest pair has one point in each set. Therefore, a divide and conquer approach would *not* necessarily satisfy the optimal substructure property.

3. **Randomization**

- 3a.

We know that $E[Y(n)] < 2nlnn = O(nlogn)$

We can plug this into Markov's inequality

Let Y(n) be the random number of comparisons quicksort makes on an array of size n.

This means that: $Y(n) => cn^2$

This means we want a bound of $P[Y(n) => cn^2]$

Markov's inequality: for any non-negative random variable X and a>0

$P[X => a] <= E[X]/a$

Take $X = Y(n)$ and $a = cn^2$

$P[Y(n) => cn^2] <= E[Y(n)]/cn^2 <= 2nlnn/cn^2 = 2lnn/cn$

The probability is <= O(logn/n).

- 3b.

Let Y(n) be the random number of comparisons quicksort makes on an array of size n.

$E[Y(n)] <= 2nlnn$

We want $P[Y(n) => 10^c nlnn]$

Using Markov's inequality:

$P[Y(n) => 10^c nlnn] <= E[Y(n)]/10^c nlnn <= 2nlnn/10^c nlnn = 2/10^c$

4. **Greedy Algorithms**

Greedy Choice Property: for any set of jobs, there exists an optimal schedule that begins with the job of smallest processing time.

Proof: Let A be the set of jobs, G the greedy schedule that starts with the shortest job, and O some optimal schedule.

If O already starts with the shortest job, we are done. Otherwise, let j be the shortest job and suppose O begins with some job k!=j such that $p_k > p_j$

Consider the first occurence of these two jobs in O, and suppose O contains the pair (k,j) in that order.

We construct schedule O' by swapping these two jobs.

The waiting time of all other jobs remains unchanged, the only change is:

- In O
  - k waits T time
  - j waits T + p_k
- In O':
  - j waits T time
  - k waits T + p_j

Therefore:

Total in $O = 2T + p_k$

Total in $O' = 2T + p_j$

Since $p_k > p_j$, the new schedule has lower total waiting time, contradicting the optimality of O. Therefore, O must start with job j, proving that the greedy choice is always part of some optimal schedule.

5. **Dynamic Programming**

Maximum span (no parallelism):

0–1 Knapsack, from dp-01.

The dependency structure reduces the capacity by 1 at each recursive step, giving a longest path of length n, so the span is O(n).

Polylogarithmic span (ideal parallelism):

Optimal Binary Search Tree (OBST), from dp-03.

Computing the minimum root candidate for each interval can be done in parallel with O(logn) span per interval, giving total span O(nlogn).

6. **Graphs**

Let T be the minimum spanning tree of G.

Suppose, for contradiction, that T contains the largest weight cycle C.

1. remove e from T. Since T is a tree, removing any edge disconnects it into two components. We can call the resulting sets A and B.
2. because e lies on the cycle C, there is another path in C between the same endpoints that avoids e. The path must contain at least one other edge f that goes from A to B (other wise we couldn't get from A to B)
3. all edges on C have distinct weights and e is the heaviest on C, so w(f)< w(e)
4. now consider T'=T-{e}+{f}
   - T' is still a spanning tree (we removed one ege and added another one that reconnect the two components)

- the total weight is now

$w(T')=w(T)-w(e)+w(f) < w(T)$

So we have constructed a spanning tree lighter than T, contradicting the assumption that T was a minimum spanning tree. Therefore, our assumption was false.