CS 161 Fall 2017: Section 8 Solutions

Fair Division

You have a cake sliced into n differently sized pieces. You need to split it among k students in your class where $k \leq n$ and you want to give each student exactly 1 piece of cake. However, you know that since the pieces are differently sized, the students who end up with smaller pieces will most likely complain about *unfairness*. You want to minimize the complaints by minimizing *unfairness*. If you select k cake pieces of size $S_1, ..., S_k$ then unfairness is defined as $max(S_1, ..., S_k) - min(S_1, ..., S_k)$.

(a) Describe a very simple algorithm to maximize unfairness.

Take the largest and smallest values. This guarantees the largest unfairness value because we have the largest difference we could possibly generate with the pieces we have.

(b) Describe an algorithm to minimize unfairness.

1. Sort the pieces by size.
2. Slide a window of size k over the sorted array, and check the difference between the largest and smallest values in the window.
3. Return the window with the smallest difference.

Restricted MST

Given an undirected weighted graph $G = (V, E)$, we have a set $U \subset V$. We wish to find a minimum spanning tree such that all nodes in U are leaf nodes. (The result may not be a MST of the original graph G.)

Define $T = V - U$. Create an MST out of the nodes in T (using Kruskal's algorithm for example). Then, add nodes in U to this tree by taking the lightest edge from a node $u \in U$ to another node $t \in T$.

Pareto Points

Given a set of 2d points P, a Pareto optimal point is a point (x, y) such that $x > x'$ or $y > y'$, $\forall (x', y') \neq (x, y) \in P$. Develop an algorithm to find all Pareto optimal points.

```
Data: list of points $(x_n, y_n)$
sort input points (largest first) by x coordinate, then tiebreak by y
initialize pareto points array with first point from the sorted list
set $Y$ to y coordinate of that point
for point in sorted input do
    if y coordinate of point $> Y$ then
        add point to pareto points array
        update $Y$ to the y coordinate of this point
return pareto points array
```

This algorithm works in $O(n \log n)$ because sorting the input takes $n \log n$, and all other operations are linear.

This algorithm works as follows: Once we have the points sorted by x, then the largest point is obviously a pareto point (everything else has either smaller x coordinates, and if they have equal x coordinates, then our sort indicates that they have smaller y coordinates).

Now, as we progress down our input list, each subsequent point has either equal or smaller x coordinates. Therefore, the only way that it could possibly be a pareto point is if it has a larger y coordinate. Therefore, we save the value of the largest y coordinate we have yet encountered, and a pareto point lower in the list must have a larger y coordinate than our saved value.