CSCI 270 Lecture 18: Divide and Conquer

All Pairs Shortest Paths Strikes Back

We want to find the shortest path between all pairs of points (we'll return $n(n-1)$ different answers, one for each pair).

Let $ASP[i, x, z]$ = the length of shortest path from x to z using no more than i edges.

$ASP[i, x, x] = 0$
$ASP[0, x, z] = \infty$
$ASP[i, x, z] = \min_{(x,y)\in E}(c_{(x,y)} + ASP[i-1, y, z])$

```
For i = 0 to n − 1
    For all nodes z
        For all nodes x
            Calculate ASP[i, x, z]
```

We can argue the last loop takes $\theta(m)$ time, and there are n iterations for the other loops, so the runtime is $\theta(mn^2)$.

- If we're at node x and we need to get to node z, the dynamic programming way is to find the next node after x that we visit. What would the divide and conquer way be?
- Using this idea, what would our new recursive formula be?
- What values of i do we need to iterate over?
- What would our runtime for this algorithm be?

Integer Multiplication

1. Elementary Math time! How do you calculate $12 \cdot 13$?
2. How would a computer calculate it?
3. What is the running time to multiply two n-bit integers?

We're going to try to use Divide and Conquer to improve on this.

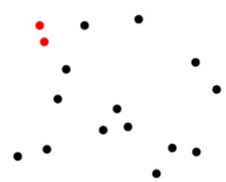
Let $x = x_F \cdot 2^{\frac{n}{2}} + x_L$, where x_F is the first $\frac{n}{2}$ bits of x. Similarly, $y = y_F \cdot 2^{\frac{n}{2}} + y_L$.

$xy = x_F y_F 2^n + (x_F y_L + x_L y_F)2^{\frac{n}{2}} + x_L y_L$

1. What's our base case?
2. What is the recurrence relation here?
3. What would our runtime be?
4. What part of our algorithm needs improvement?
5. How can we improve it? Hint: What is $(x_F + x_L)(y_F + y_L)$?

Closest Points on a Plane

Given n points on a plane (specified by their x and y coordinates), find the pair of points with the smallest euclidean distance between them.



1. What runtime can you achieve simply using brute-force?
2. Using Divide and Conquer, how should we divide the plane?
3. What do we need to do in our combine phase?
4. What's the recurrence relation for our algorithm?
5. What runtime does this achieve?

Let δ be the min distance between any pair so far. Instead of comparing all pairs of points on each side, we will instead only look at points within δ of the boundary.

What's the worst-case runtime for our new algorithm?

Instead of comparing all pairs of points within δ of the boundary, we will only compare points if their y-coordinates are within δ of each other.

What's the worst-case runtime now?