7 Greedy Algorithms, Shortest Paths

- Greedy algorithms (making change, shortest paths, minimum spanning trees)
- Up next: scheduling
- Gear demo from class.

7.1 Greedy Algorithms

We make a sequence of choices, each of which is locally as good as possible.

Example 7.1
Making change.

Fix some n different coin denominations in our currency system: $c_1, c_2, \cdots, c_n$.
We are given a value v, we want to find a way to make v using as few coins as possible.

Our greedy algorithm:

```
Change(v):
    if v = 0, return
    else
        let c be the largest coin of denomination <= v
        add c to the list of coins
        Change(v - c)
```

An example of our greedy algorithm not working correctly: Suppose we have coins 1, 3, and 4.
Change(6) outputs $4, 1, 1$, when $6 = 3 + 3$ is the optimal solution to our problem.

Another example is if we have coins $1, 5, 10, 25, 100$, then our greedy algorithm will always produce the minimum number of coins.

7.2 Shortest paths

BFS finds shortest paths in unweighted graphs. What if there is a weight w_{uv} for each edge$(u, v) \in E$?
Here, the length of the path is the sum of the edge weights.

One way to do this is by replacing integer valued weights with an equivalent number of unweighted in between two nodes, but this only works well in certain situations.

Alternatively, we can use Djikstra's algorithm.

```
Dijkstra(G, w, s):
    let d[s] = 0 and pr[s] = null
    for all v in V where v is not s, let d[v] = infty
    mark all vertices as unexplored
    repeat:
        let u be the unexplored vertex with smallest d[u]
        for each neighbor v of u
            if d[u] + w_uv < d[v] then
                let d[v] = d[u] + w_uv
                let pr[v] = u
            endif
        endfor
        mark v as explored
    until all vertices are explored.
```

Why is this algorithm correct?

Lemma 7.2
When a vertex v is explored, $d[v]$ is the distance from s to v.

Proof. By induction on the number of explored vertices.

Clearly the property holds when only s is explored (base case)

Suppose the lemma holds when there are k explored vertices.
Let v be the $k+1$st explored vertex, and let $pr[v] = u$.

Suppose for a contradiction that the shortest path from from s to v does not go through u.

Instead, let $x \neq u$ be the last explored vertex on the shortest path and let y be its unexplored neighbor on this path.

Note that $y \neq v$ since otherwise the algorithm would choose $u = x$.

Then, $\text{dist}(s, v) < d[v] \leq d[y]$ because we chose to add v before y. Here, $d[y]$ is the length of a path from s to y found by the algorithm. $\leq \text{dist}(s, v)$ since y is on the shortest path from s to v.

This is a contradiction, so the shortest path from s to v must go through u, and $dist(s, v) = d[u] + w_u v = d[v]$. □