Goals:
- Practice designing a DP algorithm

Announcements:
- Prof. Linderman Today! 4:30 Hillcrest 103
- Friday - discuss Search and Dr. Noble's research
- Monday @7 on Zoom: Tapia Panel

Recurrence Relation for $S_{i,r}$

$$S_{i,r} = \begin{cases} S_{i-1,r} & \text{if } i \notin S_{i,r} \\ S_{i-1,r-c_i} \cup \{i\} & \text{if } i \in S_{i,r} \end{cases}$$

$\forall r \; S_{0,r} = \emptyset$ Base Case

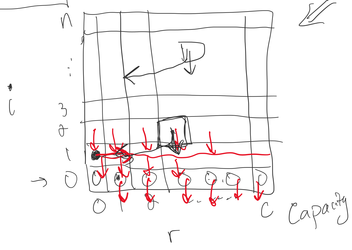
Recurrence Relation for Objective Function

$$V(S_{i,r}) = \max \{ V(S_{i-1,r}), V(S_{i-1,r-c_i}) + v_i \}$$

$\forall r, V(S_{0,r}) = 0$

Only an option if $r \geq c_i$

$A \Rightarrow A[2,3] = V(S_{2,3})$



Pseudocode

Input: $n \times 1$ arrays V (values) and C (costs)
Output: Optimal set to put in knapsack

```
Initialize A as (n+1) x (C+1) array
//Base Case

For r = 0 to C
    A[0,r] = 0

For i = 1 to n:
    For r = 0 to C:
        if r ≥ c_i:
            A[i,r] = max{ A[i-1,r], A[i-1, r-c_i] + v_i }
        else:
            A[i,r] = A[i-1,r]
```

Test

Ex: $C = 5$

Item	Value	Cost
1	6	2
2	5	1
3	8	3
4	7	4

Q: What is the runtime of DP knapsack algorithm?

A) $O(n)$ B) $O(C)$ C) $O(n+C)$ D) $O(nC)$