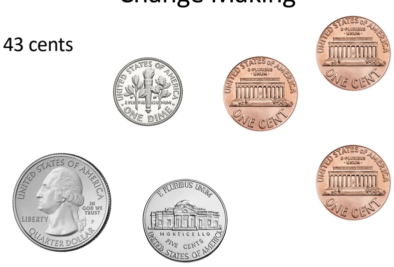
## CS4102 Algorithms
Spring 2019

**Warm up**

Given access to unlimited quantities of pennies, nickels dimes, and quarters, (worth value 1, 5, 10, 25 respectively), provide an algorithm which gives change for a given value $x$ using the fewest number of coins.

## Change Making

43 cents



## Change Making Algorithm

- Given: target value $x$, list of coins $C = [c_1, \ldots, c_n]$
  (in this case $C = [1,5,10,25]$)
- Repeatedly select the largest coin less than the remaining target value:

while($x > 0$)
  let $c = \max(c_i \in \{c_1, \ldots, c_n\} \mid c_i \leq x)$
  print $c$
  $x = x - c$

## Why does this always work?

- If $x < 5$, then pennies only
  - 5 pennies can be exchanged for a nickel
    Only case Greedy uses pennies!
- If $5 \leq x < 10$ we must have a nickel
  - 2 nickels can be exchanged for a dime
    Only case Greedy uses nickels!
- If $10 \leq x < 25$ we must have at least 1 dime
  - 3 dimes can be exchanged for a quarter and a nickel
    Only case Greedy uses dimes!
- If $x \geq 25$ we must have at least 1 quarter
  Only case Greedy uses quarters!

4

**Warm up**

Given access to unlimited quantities of pennies, nickels dimes, mays, and quarters, (worth value 1, 5, 10, 11, 25 respectively), give 90 cents change using the fewest number of coins.


5

## Greedy solution

90 cents


6

## Greedy solution

90 cents



## Today's Keywords

- Greedy Algorithms
- Choice Function
- Change Making
- Interval Scheduling
- Exchange Argument



## CLRS Readings

- Chapter 16

## Homeworks

- Homework 5 due tonight at 11pm
  - Seam Carving!
  - Dynamic Programming (implementation)
  - Java or Python
- Homework 6 out tonight, due Wednesday April 3 at 11pm
  - Dynamic Programming and Greedy Algorithms
  - Written (using Latex!)

10

## Greedy vs DP

- Dynamic Programming:
  - Require Optimal Substructure
  - Several choices for which small subproblem
- Greedy:
  - Require Optimal Substructure
  - Must only consider one choice for small subproblem

11

## Greedy Algorithms

- Require Optimal Substructure
  - Solution to larger problem contains the solution to a smaller one
  - Only one subproblem to consider!
- Idea:
  1. Identify a greedy choice property
     - How to make a choice guaranteed to be included in some optimal solution
  2. Repeatedly apply the choice property until no subproblems remain

12

## Change Making Choice Property

- Largest coin less than or equal to target value must be part of some optimal solution (for standard U.S. coins)

13

## Interval Scheduling

- Input: List of events with their start and end times (sorted by end time)
- Output: largest set of non-conflicting events (start time of each event is after the end time of all preceding events)

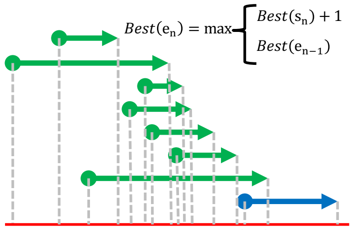
| | |
|---|---|
| [1, 2.25] | Alumni Lunch |
| [3, 4] | CHS Prom |
| [3.5, 4.75] | CS4102 |
| [4, 5.25] | Bingo |
| [4.5, 6] | SCUBA lessons |
| [5, 7.5] | Roller Derby Bout |
| [7.75, 11] | UVA March Madness watch party |

14

## Interval Scheduling DP

$Best(t)$ = max # events that can be scheduled before time $t$

$$Best(\mathrm{e_n}) = \max\begin{cases} Best(\mathrm{s_n}) + 1 & \text{Include event } n \\ Best(\mathrm{e_{n-1}}) & \text{Exclude event } n \end{cases}$$




15

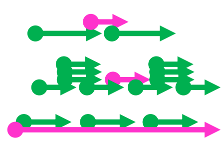
## Greedy Interval Scheduling

- Step 1: Identify a greedy choice property
  - Soonest starting event
  - Soonest ending event
  - shortest event

16

## Greedy Interval Scheduling

- Step 1: Identify a greedy choice property
  - Options:
    - Shortest interval
    - Fewest conflicts
    - Earliest start
    - Earliest end

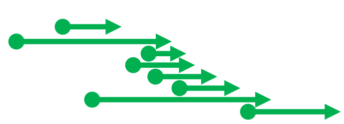


Prove using Exchange Argument

17

## Interval Scheduling Algorithm

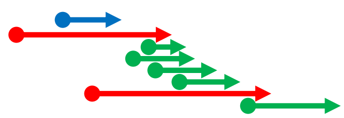
Find event ending earliest, add to solution,
Remove it and all conflicting events,
Repeat until all events removed, return solution



18

## Interval Scheduling Algorithm

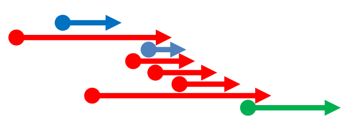
Find event ending earliest, add to solution,
Remove it and all conflicting events,
Repeat until all events removed, return solution



19

## Interval Scheduling Algorithm

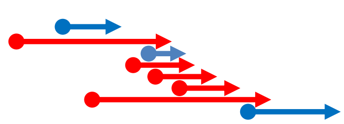
Find event ending earliest, add to solution,
Remove it and all conflicting events,
Repeat until all events removed, return solution



20

## Interval Scheduling Algorithm

Find event ending earliest, add to solution,
Remove it and all conflicting events,
Repeat until all events removed, return solution



21

## Interval Scheduling Run Time

Find event ending earliest, add to solution,
Remove it and all conflicting events,
Repeat until all events removed, return solution

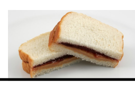
```
Equivalent way

StartTime = 0
For each interval (in order of finish time):            O(n)
        if begin of interval < StartTime or end of interval < StartTime:     O(1)
                do nothing
        else:
                add interval to solution            O(1)
                StartTime = end of interval
```

22

## Exchange argument

- Shows correctness of a greedy algorithm
- Idea:
  - Show exchanging an item from an arbitrary optimal solution with your greedy choice makes the new solution no worse
  - How to show my sandwich is at least as good as yours:
    - Show: "I can remove any item from your sandwich, and it would be no worse by replacing it with the same item from my sandwich"



23

## Exchange Argument for Earliest End Time

- Claim: earliest ending interval is always part of some optimal solution

- Let $OPT_{i,j}$ be an optimal solution for time range $[i, j]$
- Let $a^*$ be the first interval in $[i, j]$ to finish overall
- If $a^* \in OPT_{i,j}$ then claim holds
- Else if $a^* \notin OPT_{i,j}$, let $a$ be the first interval to end in $OPT_{i,j}$
  - By definition $a^*$ ends before $a$, and therefore does not conflict with any other events in $OPT_{i,j}$
  - Therefore $OPT_{i,j} - \{a\} + \{a^*\}$ is also an optimal solution
  - Thus claim holds

24

25