

# Greedy Algorithms

Mattox Beckman

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN  
DEPARTMENT OF COMPUTER SCIENCE

Spring 2023

# Objectives

- ▶ Describe the characteristics of a greedy algorithm
- ▶ Show how to use a greedy algorithm to solve several classic problems

# Properties of Greedy Algorithms

1. They have *optimal substructure* — subproblems have optimal solutions that can be

combined to get the main solution.

1. They have the *Greedy Property* — We will never regret making a greedy choice locally.

# Classic Example: Coin Change

- ▶ Given coins of values 25, 10, 5, 1: make 57 with as few coins as possible.
- ▶ This version can be solved greedily!
  - ▶  $57 = 25 \times 2 + 5 + 1 \times 2$ .

```
1  int numCoinTypes, amount, count, i;
2  cin >> numCoinTypes;
3  vi coins;
4  for(i=0; i<numCoinTypes; ++i) {
5      cin >> x;  coins.push_back(x);
6  }
7  cin >> amount;
8  count = 0; i=0;
9  while (amount > 0)
10     if (coins[i] <= amount) {
11         amount -= coins[i]; ++count;
12     }
```

# Coin change is not always greedy

- ▶ Suppose we have coin values 25, 20, 5, 1.

# Coin change is not always greedy

- ▶ Suppose we have coin values 25, 20, 5, 1.
- ▶ What is the optimal way to make 40 cents change now?

# Coin change is not always greedy

- ▶ Suppose we have coin values 25, 20, 5, 1.
  - ▶ What is the optimal way to make 40 cents change now?
- ▶ Greedily:  $25 + 5 + 5 = 3$  coins

# Coin change is not always greedy

- ▶ Suppose we have coin values 25, 20, 5, 1.
  - ▶ What is the optimal way to make 40 cents change now?
- ▶ Greedily:  $25 + 5 + 5 = 3$  coins
- ▶ Optimal:  $20 \times 2$



# Classic Example: Activity Selection Problem

- ▶ Given a list of activities with start and finish times, what is the maximum number of activities someone can do?

# Classic Example: Activity Selection Problem

- ▶ Given a list of activities with start and finish times, what is the maximum number of activities someone can do?
  - ▶ Assume only one activity at a time.

# Classic Example: Activity Selection Problem

- ▶ Given a list of activities with start and finish times, what is the maximum number of activities someone can do?
  - ▶ Assume only one activity at a time.
- ▶ Sort activities by finish times

# Classic Example: Activity Selection Problem

- ▶ Given a list of activities with start and finish times, what is the maximum number of activities someone can do?
  - ▶ Assume only one activity at a time.
- ▶ Sort activities by finish times
- ▶ Add first activity to list

# Classic Example: Activity Selection Problem

- ▶ Given a list of activities with start and finish times, what is the maximum number of activities someone can do?
  - ▶ Assume only one activity at a time.
- ▶ Sort activities by finish times
- ▶ Add first activity to list
- ▶ Repeat: take first activity that has start time after last finish time.

# Source Code

- ▶ Assume `a` has pairs representing the activities.

```
1  vii a; // actvitiy pairs
2  int last;
3  cout << a[0] << endl;
4  last = a[0].second;
5  for(i=1; i<a.length; ++i)
6      if (a[i].first >= last) {
7          cout << a[i] << endl;
8          last = a[i].second;
9      }
```

# In contests

- ▶ Use it if you can, but be sure. Otherwise, use Complete Search or DP.
- ▶ Learn a few classic algorithms: coin change, load balancing, interval covering
- ▶ Preprocessing input can help... e.g., sorting your input first.