# Greedy Algorithms

# Making Change

A vending machine stocks pennies, nickels, and quarters. What is the fewest number of coins the vending machine must dispense to return exactly **N** cents to the customer?

# Making Change

Key observations:

- 1 quarter == 5 nickels == 25 pennies
- 1 nickel == 5 pennies

No downside to using larger coin whenever possible

# Making Change

quarters = N/25

N = N – 25*quarters

nickels = N/5

N = N – 5*nickels

pennies = N

return quarters + nickels + pennies

# Greedy Algorithms

**Local optimality**: the best single move you can make right now

- (dispense the biggest coin <= **N**)

# Greedy Algorithms

**Local optimality**: the best single move you can make right now

- (dispense the biggest coin <= **N**)

**Global optimality**: the best long-term move

# Greedy Algorithms

Best-case scenario:

local optimality --> global optimality

"No Regrets" principle: every good decision now will still stay a good decision later

# Making Change II

A vending machine stocks pennies, dimes, and quarters. What is the fewest number of coins the vending machine must dispense to return exactly **N** cents to the customer?

# Making Change II

No Regrets principle **not** satisfied

- 30 = 25 + 1 + 1 + 1 + 1 + 1 (6 coins)
- 30 = 10 +10 + 10 (3 coins, optimal)

# Making Change II

No Regrets principle **not** satisfied

- 30 = 25 + 1 + 1 + 1 + 1 + 1 (6 coins)
- 30 = 10 +10 + 10 (3 coins, optimal)

Cannot use greedy algorithm, must sometimes backtrack

- (dynamic programming)

# Identifying Greedy Solutions


Before coding: **is the greedy local choice globally optimal?**

# Identifying Greedy Solutions

Typical reasoning structure:

"Suppose you have a global solution. Switching choice **k** to the locally optimal solution makes the global solution better (or doesn't make it worse)."

# Identifying Greedy Solutions

Typical reasoning structure:

"Suppose you have a global solution. Switching choice **k** to the locally optimal solution makes the global solution better (or doesn't make it worse)."

"Suppose you make change and have >= 25 cents of non-quarter change. It's always possible, and more optimal, to replace exactly 25 cents with a quarter."

# Greedy(?) Problem I

A career fair has **N** interview slots, beginning at time $\mathbf{a}_i$ and ending at $\mathbf{b}_i$. You cannot attend two interviews that overlap in time. What is the maximum number of interviews you can attend?

# Greedy(?) Problem I

Is there a no-regrets choice?

Pick event that starts first?

# Greedy(?) Problem I

Is there a no-regrets choice?

Pick event that starts first?

- could take up the entire fair…

# Greedy(?) Problem I

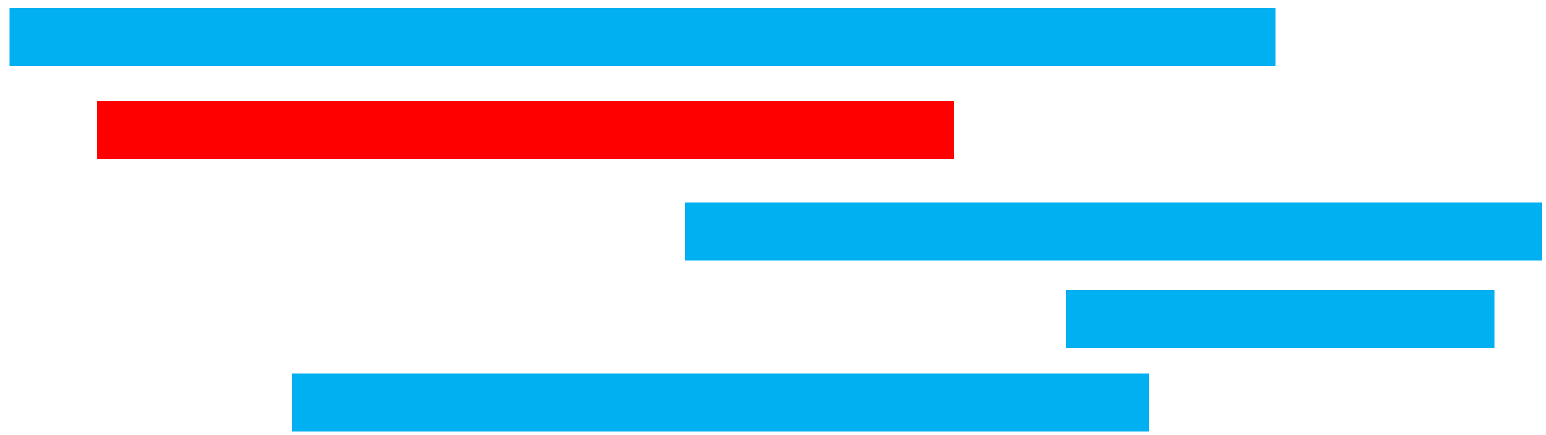
Is there a no-regrets choice?

Claim: the event that **starts last**

# Greedy(?) Problem I

Any schedule that does not include the event starting last:

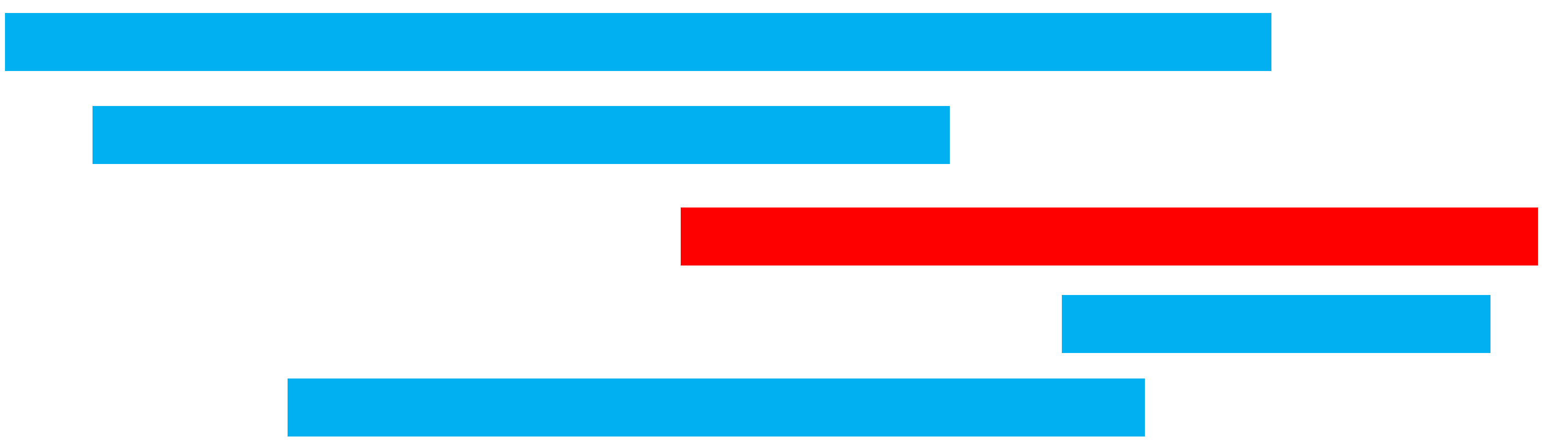
- you can improve by adding last event;



# Greedy(?) Problem I

Any schedule that does not include the event starting last:

- you can improve by adding last event;
- or by switching to last event



# Greedy(?) Problem II

You have \$**X** and can buy any subset of N items costing \$$\mathbf{p}_i$. What is the largest set of items you can buy?

Example: **X** = 100

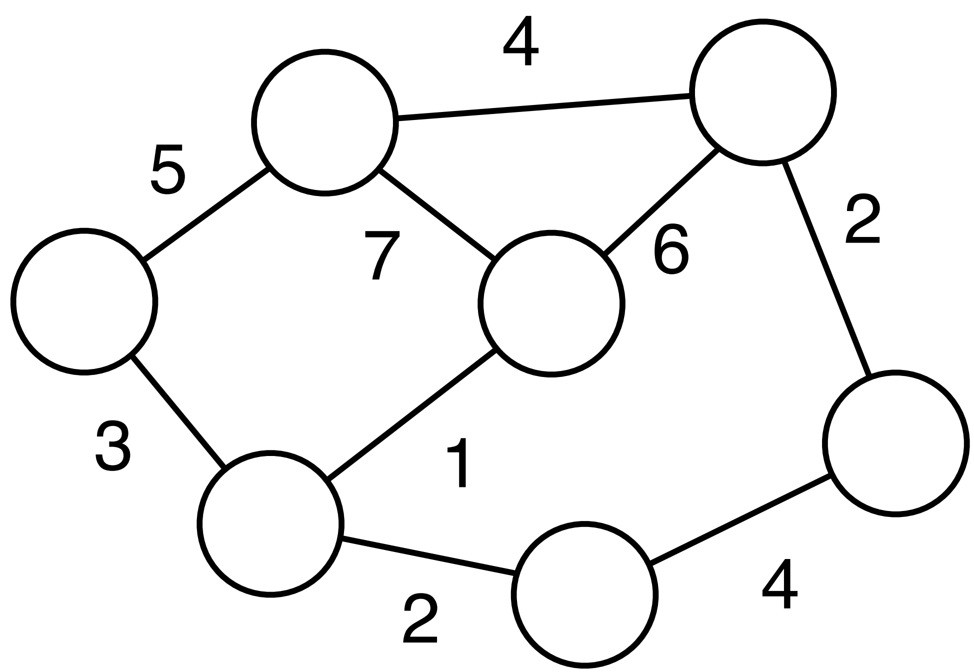
**p** = {1, 20, 30, 50, 90}

# Greedy(?) Problem III

Google is connecting your neighborhood with fiber. The cost of connecting each pair of houses with fiber is $\mathbf{c}_{ij}$.

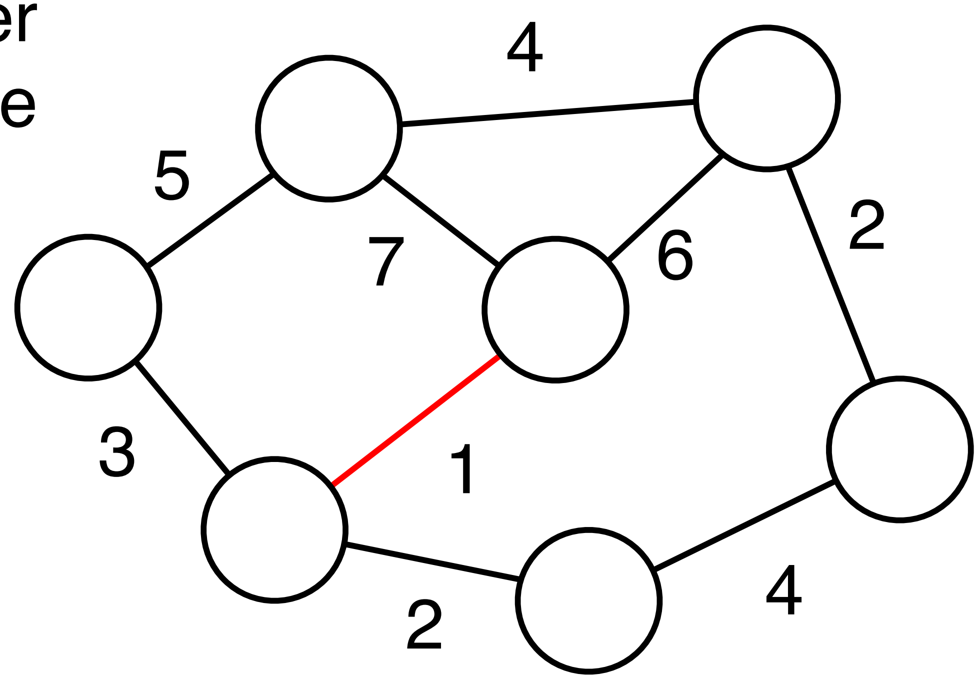
What is the least Google must pay for a network that ensure that a path of fiber exists from any house to any other house?

# Greedy(?) Problem III



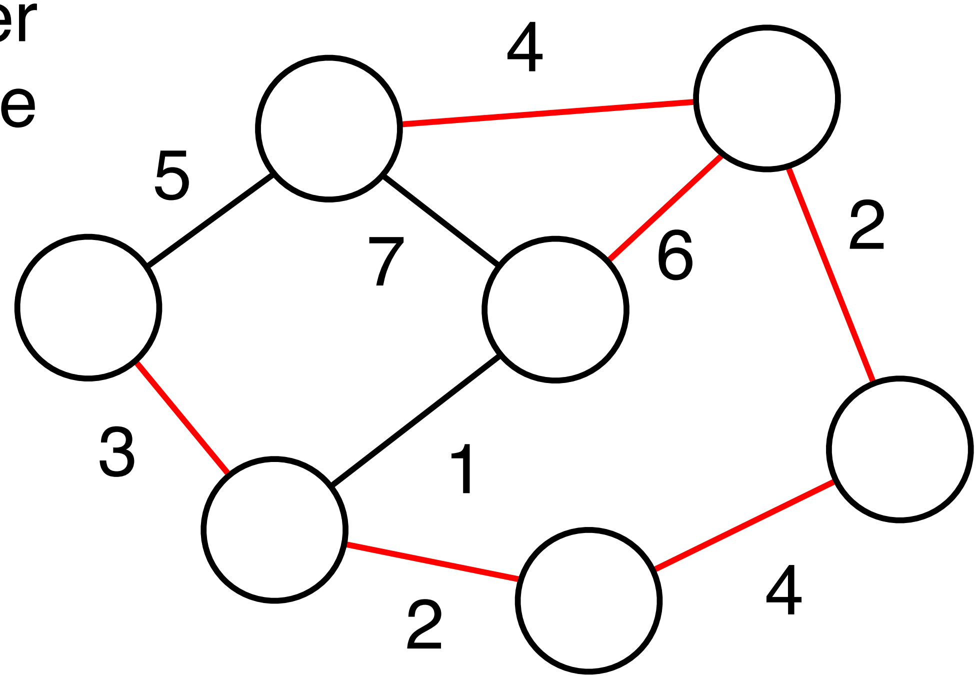
# Greedy(?) Problem III

Claim: build fiber on shortest edge



# Greedy(?) Problem III

Claim: build fiber on shortest edge

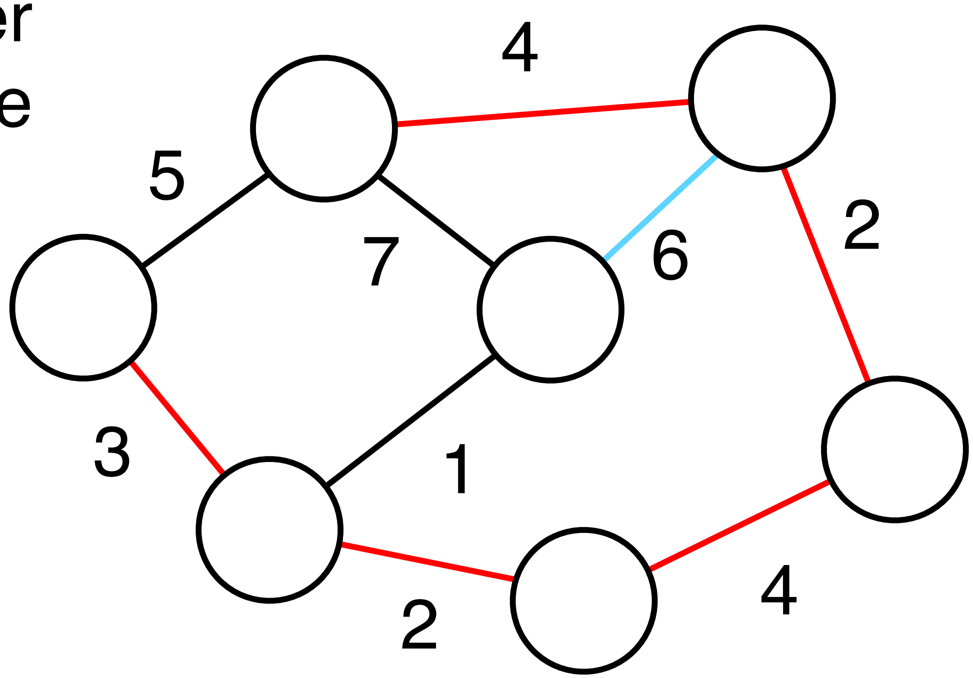



For any network not including the shortest edge,

…

# Greedy(?) Problem III

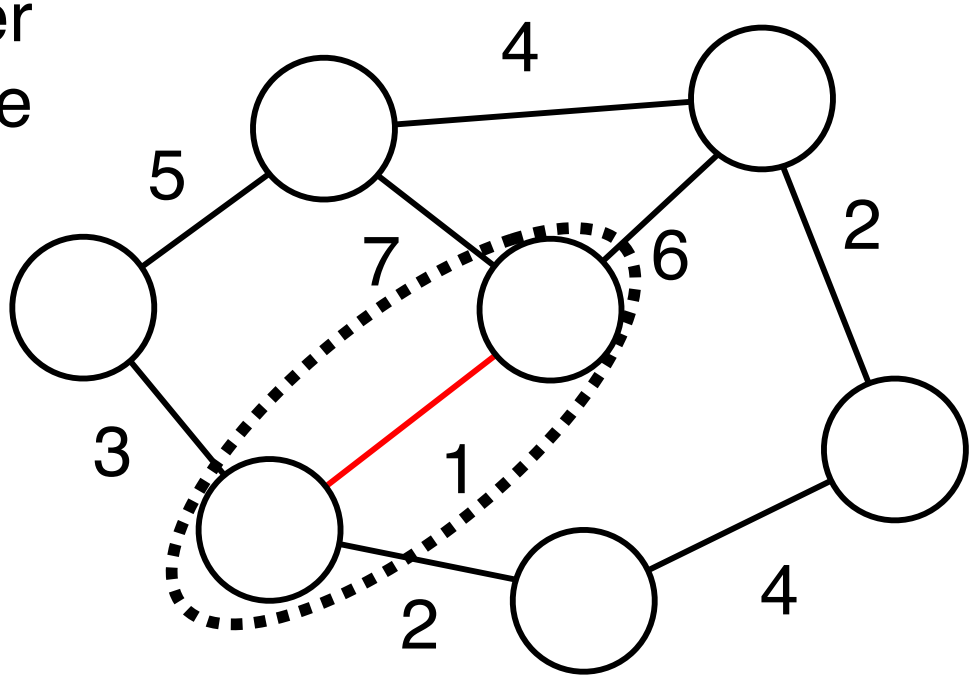
Claim: build fiber on shortest edge




For any network not including shortest edge, can improve by swapping in edge

# Greedy(?) Problem III

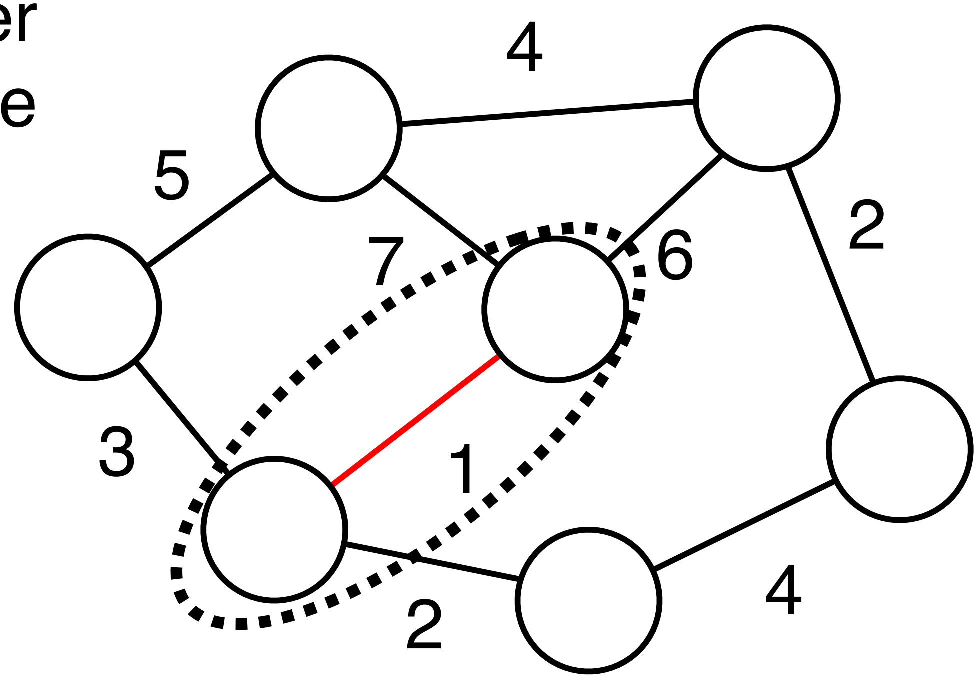
Claim: build fiber on shortest edge




Now group connected nodes and repeat

# Prim's Algorithm (MST)

Claim: build fiber
on shortest edge




Now group connected nodes and repeat