EEL 4837
Programming for Electrical Engineers II

Ivan Ruchkin
Assistant Professor
Department of Electrical and Computer Engineering
University of Florida at Gainesville
iruchkin@ece.ufl.edu
http://ivan.ece.ufl.edu

Greedy Algorithms

Readings:

- Weiss 10.1
- Horowitz 4
- Cormen 16

Short list of Algorithm Designs

- Brute force algorithms
- Simple recursive algorithms
- Divide and conquer algorithms
- **Greedy algorithms**
- Dynamic programming algorithms
- Backtracking algorithms
- Branch and bound algorithms

Greedy Algorithm Basics

- An optimization problem is one in which you want to find, not just *a* solution, but the *best* solution

Greedy Algorithm Basics

- An optimization problem is one in which you want to find, not just *a* solution, but the *best* solution
- A "greedy algorithm" sometimes works well for optimization problems
- A greedy algorithm works in phases. At each phase:
  - You take the best you can get right now, without regard for future consequences
  - You hope that by choosing a *local* optimum at each step, you will end up at a *global* optimum

The "**greedy choice**" property of problems

Greedy Algorithms: Design Technique

- Dijkstra's algorithm is an example of a **greedy algorithm**
  - It goes "greedily" towards the low-cost path
- **Greedy choice:** in each step, make a decision that appears the best
  - Disregard long-term consequences
  - Often, it is a good heuristic: *local optimum* leads to *global optimum*

Greedy Algorithm: Example

- Suppose you want to count out a certain amount of money, using the fewest possible bills and coins
- A greedy algorithm would do this would be:
  At each step, take the largest possible bill or coin that does not overshoot

Greedy Algorithm: Example

- Suppose you want to count out a certain amount of money, using the fewest possible bills and coins
- A greedy algorithm would do this would be:
  At each step, take the largest possible bill or coin that does not overshoot
  - Example: To make $6.39, you can choose:
    - a $5 bill
    - a $1 bill, to make $6
    - a 25¢ coin, to make $6.25
    - A 10¢ coin, to make $6.35
    - four 1¢ coins, to make $6.39
- For US money, the greedy algorithm always gives the optimum solution

A Scheduling Problem

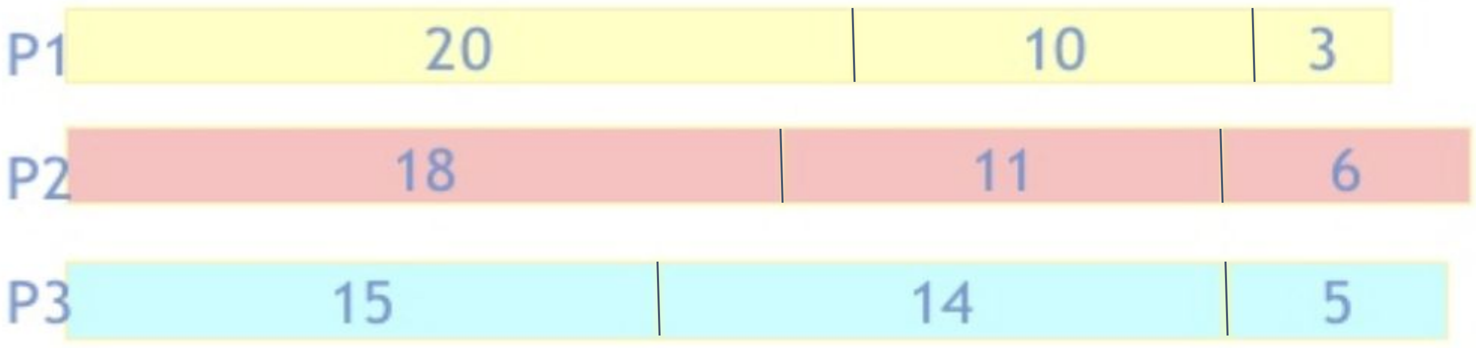
- You have to run nine jobs, with running times of 3, 5, 6, 10, 11, 14, 15, 18, and 20 minutes
- You have three processors on which you can run these jobs

A Scheduling Problem

- You have to run nine jobs, with running times of 3, 5, 6, 10, 11, 14, 15, 18, and 20 minutes
- You have three processors on which you can run these jobs
- You decide to do the longest-running jobs first, on whatever processor is available

A Scheduling Problem

- You have to run nine jobs, with running times of 3, 5, 6, 10, 11, 14, 15, 18, and 20 minutes
- You have three processors on which you can run these jobs
- You decide to do the longest-running jobs first, on whatever processor is available



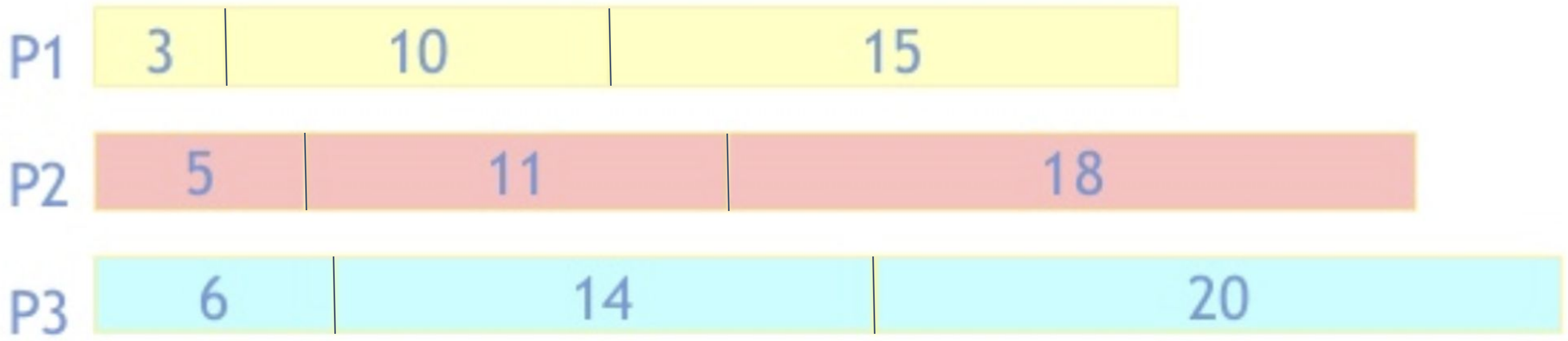
Total time to completion: 18 + 11 + 6 = 35 minutes

A Scheduling Problem

- What would be the result if you ran the *shortest* job first?
- Again, the running times are 3, 5, 6, 10, 11, 14, 15, 18, and 20 minutes

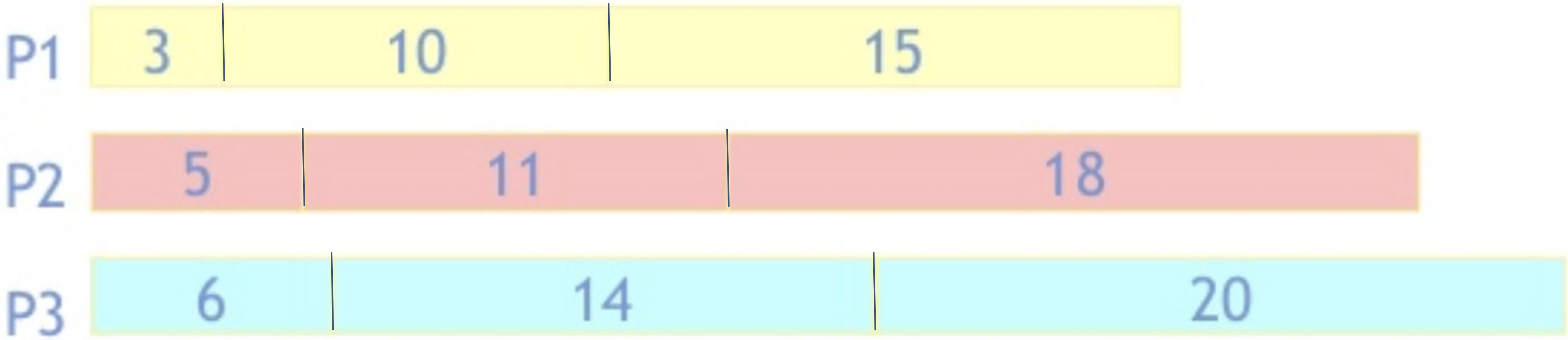
A Scheduling Problem

- What would be the result if you ran the *shortest* job first?
- Again, the running times are 3, 5, 6, 10, 11, 14, 15, 18, and 20 minutes



A Scheduling Problem

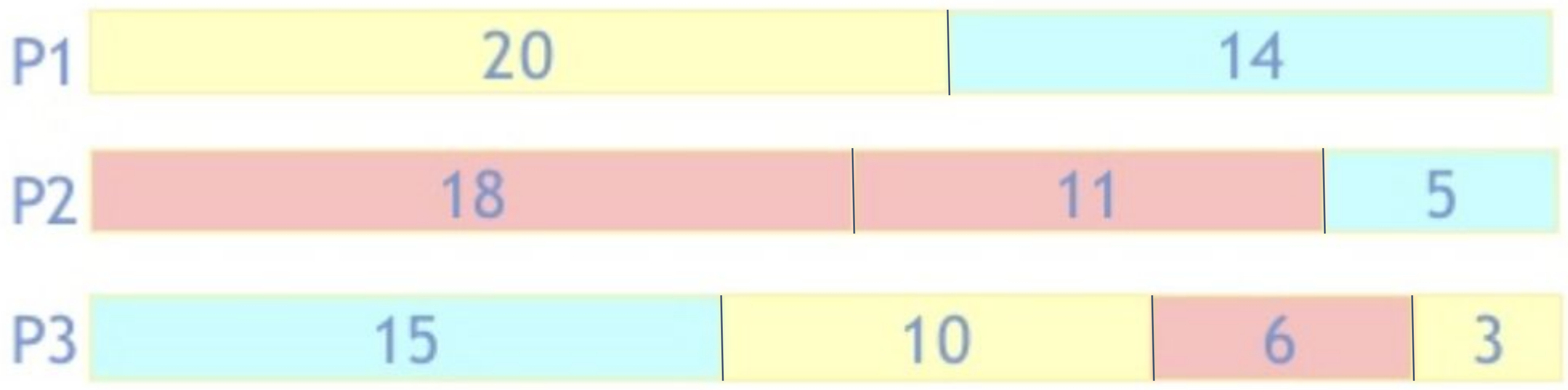
- What would be the result if you ran the *shortest* job first?
- Again, the running times are 3, 5, 6, 10, 11, 14, 15, 18, and 20 minutes



- That wasn't such a good idea; time to completion is now 6 + 14 + 20 = 40 minutes

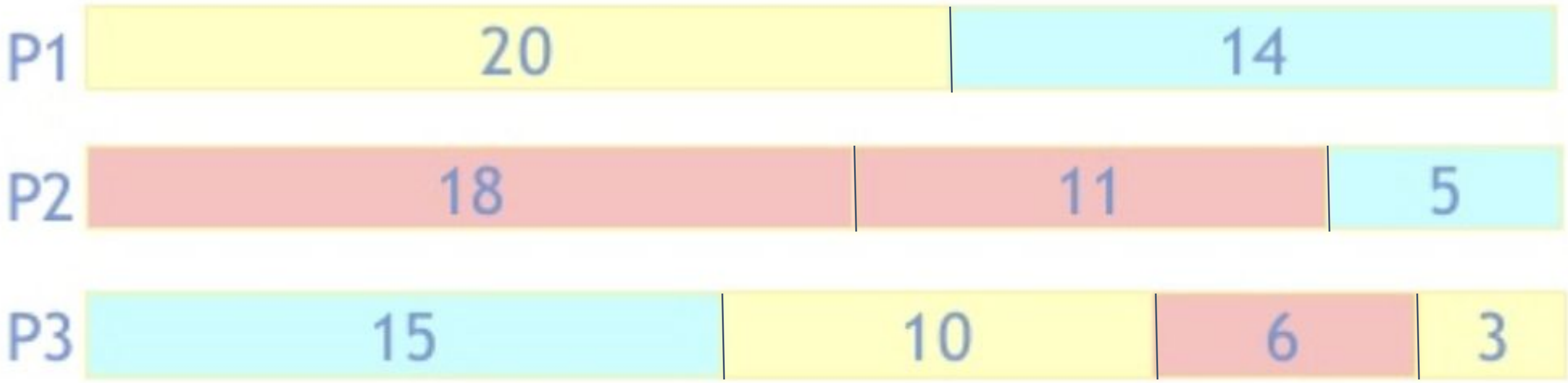
A Scheduling Problem

- Better solutions do exist:



A Scheduling Problem

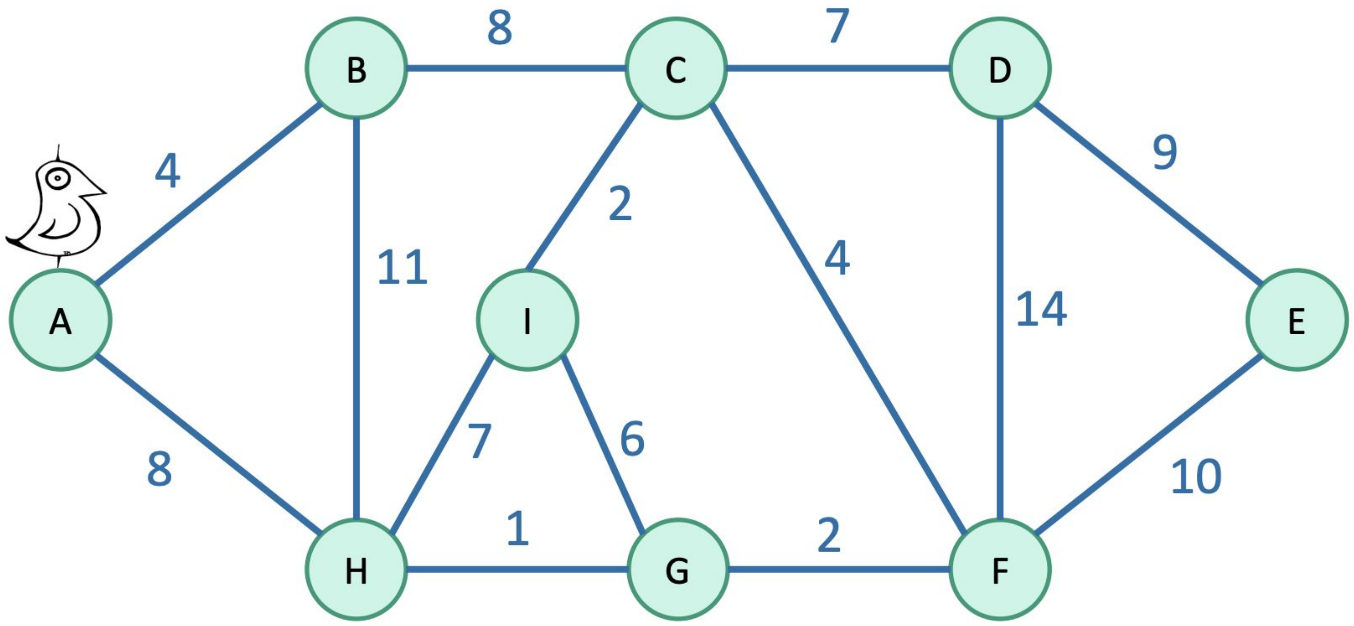
- Better solutions do exist:



- How do we find such a solution?
  - One way: Try all possible assignments of jobs to processors
  - Unfortunately, this approach can take exponential time

Prim's Algorithm

Start growing a tree, greedily add the shortest edge we can to grow the tree.



Kruskal’s Algorithm

- Sort edges by increasing edge weight
- Select the first $|V| - 1$ edges that do not generate a cycle