# Algorithms and Complexity

# Algorithmic Strategies

Lecturer: Dr. Alaa Ahmed Abbood
Lecture 7.
Class $2^{nd}$ .
Time: 8:30-10:30
Department:  Businesses Information Technology (BIT)

# Outline

- Brute-force Technique
- Greedy Technique
- Divide-and-Conquer Technique
- Homework

# Brute-force algorithms

- Straightforward way to solve a problem, based on the definition of the problem itself; often involves checking all possibilities

Properties:

- widely applicable easy
- good for small problem sizes
- often inefficient for large inputs

# Cont..

❖ Examples of problems solved by Brute force technique:

- **Selection sort:**
  - scan array to find smallest element
  - scan array to find second smallest element
  - etc.
- **Bubble sort:**
  - scan array, swapping out-of-order neighbors
  - continue until no swaps are needed
- Both take $O(n^2)$ time in the worst case
- **Sequential search:**
  - go through the entire list of $\boldsymbol{n}$ items to find the desired item
- Takes $O(n)$ time in the worst case

# Outline

- Brute-force Technique
- Greedy Technique
- Divide-and-Conquer Technique
- Homework

# Optimization problems

- 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

# Cont..

- On each step—and this is the central point of this technique—the choice made must be:
  - ***feasible***, i.e., it has to satisfy the problem's constraints.
  - ***locally optimal***, i.e., it has to be the best local choice among all feasible choices available on that step.
  - ***Irrevocable***, i.e., once made, it cannot be changed on subsequent steps of the algorithm

# Example: Counting money

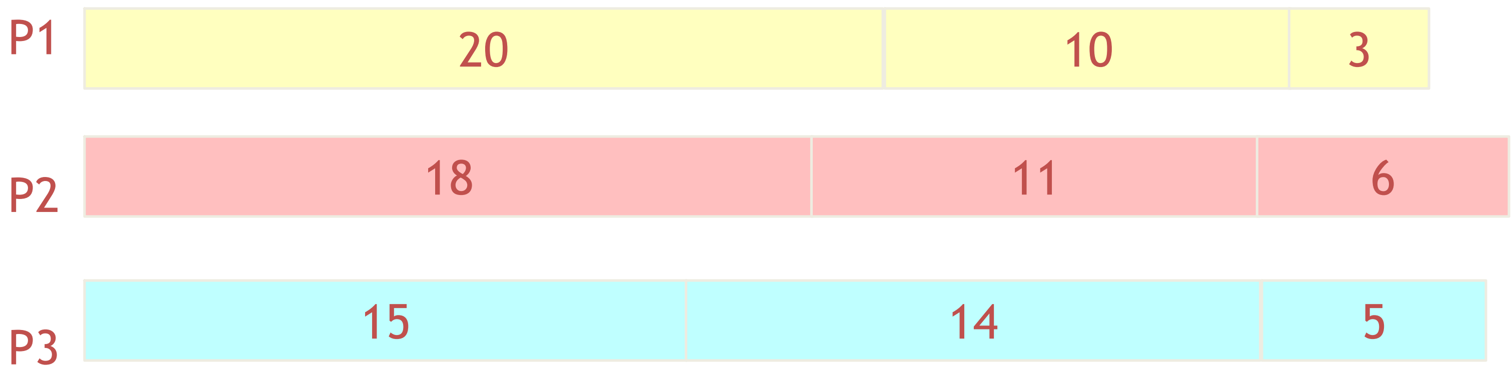
- Suppose you want to count out a certain amount of money, using the fewest possible bills and coins.
- A greedy algorithm would do this as: At each step, take the largest possible bill or coin that does not overshoot
  - Example: To make $6.39, you can choose:
    - a $5 bill
    - a $1 bill, to make $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 Failure of the greedy technique

- In some monetary system, “krons” come in 1 kron, 7 kron, and 10 kron coins
- Using a greedy algorithm to count out 15 krons, you would get A 10 kron piece
  - Five 1 kron pieces, for a total of 15 krons
  - This requires six coins
- A better solution would be to use two 7 kron pieces and one 1 kron piece
- This only requires three coins
- The greedy algorithm results in a solution, but not in an optimal solution

# A scheduling problem

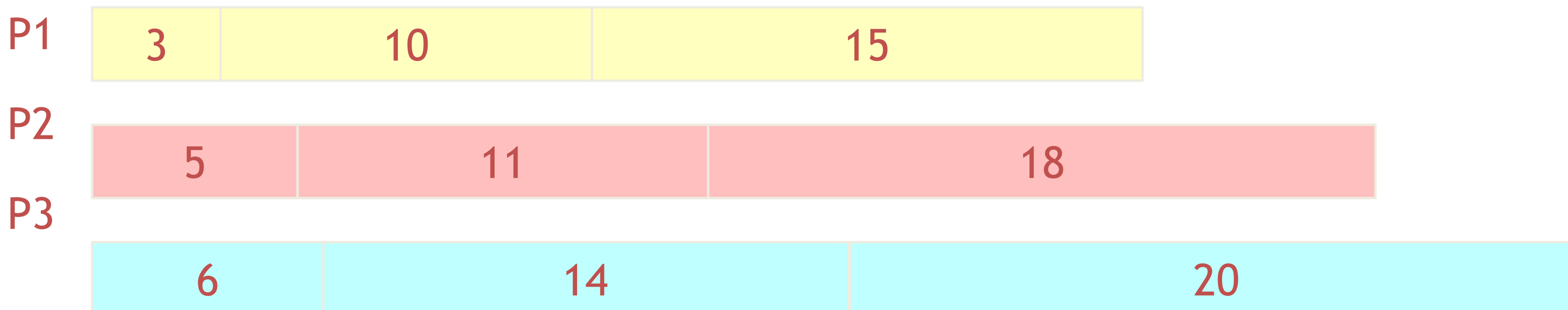
- You have to run nine jobs, with running times of 3, 5, 6, 10, 11, 14, 15, 18, and 20 minutes
- You have three processors on which you can run these jobs
- You decide to do the longest-running jobs first, on whatever processor is available




- Time to completion: 18 + 11 + 6 = 35 minutes
- This solution isn't bad, but we might be able to do better

# Another approach

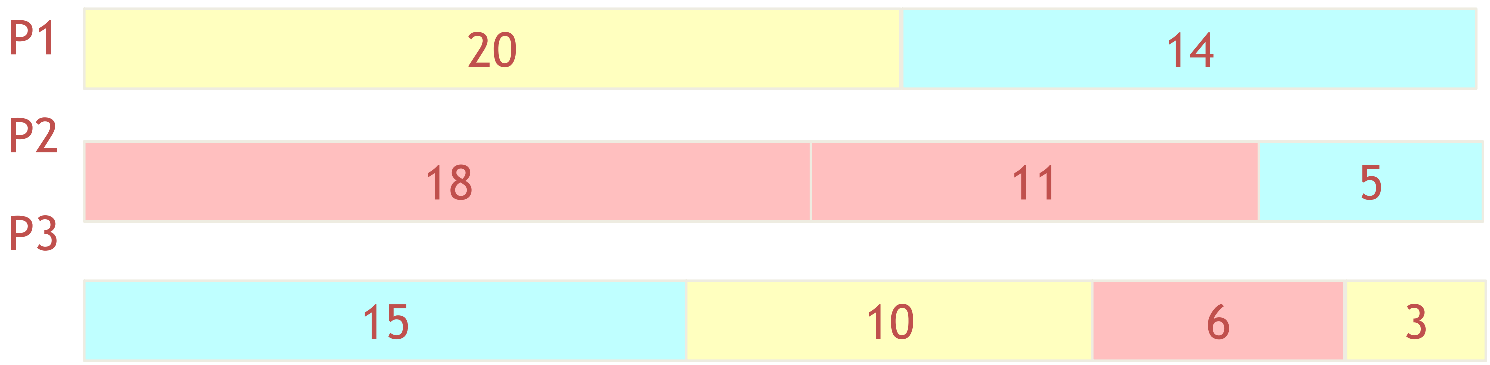
- What would be the result if you ran the *shortest* job first?
- Again, the running times are 3, 5, 6, 10, 11, 14, 15, 18, and 20 minutes




- That wasn't such a good idea; time to completion is now $6 + 14 + 20 = 40$ minutes
- Note, however, that the greedy algorithm itself is fast
  - All we had to do at each stage was pick the minimum or maximum

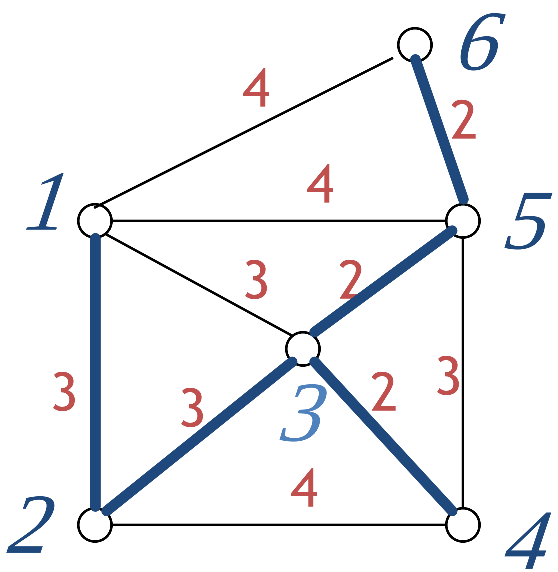
# An optimum solution

- Better solutions do exist:




- This solution is clearly optimal (why?)
- Clearly, there are other optimal solutions (why?)

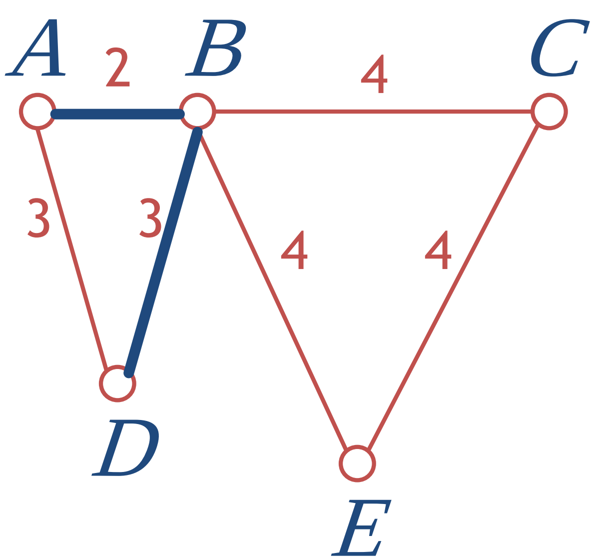
- A minimum spanning tree is a least-cost subset of the edges of a graph that connects all the nodes.
  - Start by picking any node and adding it to the tree
  - Repeatedly: Pick any *least-cost* edge from a node in the tree to a node not in the tree, and add the edge and new node to the tree
  - Stop when all nodes have been added to the tree




- The result is a least-cost (3+3+2+2+2=12) spanning tree

# Traveling salesman

- A salesman must visit every city (starting from city ***A***), and wants to cover the least possible distance
  - He can revisit a city (and reuse a road) if necessary
- He does this by using a greedy algorithm: He goes to the next nearest city from wherever he is



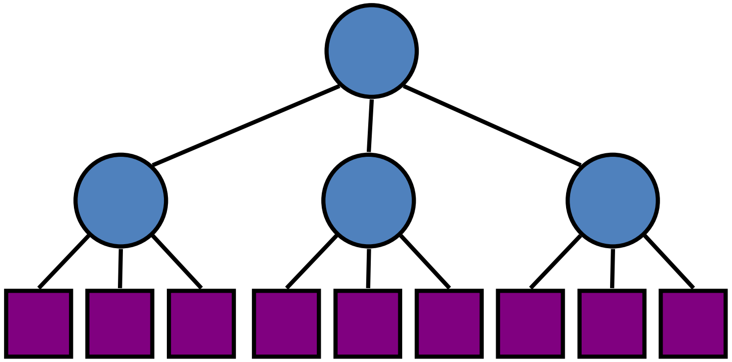

- From ***A*** he goes to ***B***
- From ***B*** he goes to ***D***
- This is *not* going to result in a shortest path!
- The best result he can get now will be ***ABDBCE***, at a cost of 16
- An actual least-cost path from ***A*** is ***ADBCE***, at a cost of 14

# Outline

- Brute-force Technique
- Greedy Technique
- Divide-and-Conquer
- Homework

# Divide-and-Conquer

- Divide-and conquer is a general algorithm design paradigm:
  - Divide: divide the input data $\boldsymbol{S}$ in two or more disjoint subsets $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, …
  - Recur: solve the subproblems recursively
  - Conquer: combine the solutions for $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, …, into a solution for $\boldsymbol{S}$
- The base case for the recursion are subproblems of constant size



# Divide and Conquer Examples

Sorting: merge sort and quicksort

Binary search

Matrix multiplication-Strassen's algorithm

# Homework

Q1// Write pseudocode for a divide-and-conquer algorithm for finding the position of the largest element in an array of *n* numbers.

# THANK YOU